

UNIVERSITY OF OSLO
Department of Informatics

Comparison of Open Source Network Intrusion Detection Systems

Jonas Taftø Rødfoss

24.may 2011



Comparison of Open Source Network Intrusion Detection Systems

Jonas Taftø Rødfoss

Network and System Administration
Oslo University College

May 24, 2011

Abstract

Many companies and organizations offer IT-services (news papers, social sites, web developers and etc.) to the public, and those services needs to be protected. The amount of computer threats are increasing drastically, and many attacks are directed to those services companies offer. Larger companies have the economy to buy expensive security tools to protect their services, while smaller companies may have the same economy.

Open source is an interesting field for those who do not have the need or the economy to buy expensive security solutions. Intrusion detection system is a well known security tool, and it could either be bought as a payment solution, or be downloaded from the web as an open source solution. Snort, Bro and Suricata are three different open source network intrusion detection systems.

By comparing installation, configuration, alarms and information one can find out which solution that fits your network best. The process of setting up the test environment, installation and configuration of Snort, Bro and Suricata, and installation of Metasploit have been a time consuming process. Snort, Bro and Suricata have been tested in a network, and against a Metasploit framework with known exploits. Running Snort, Bro and Suricata in a network, have shown huge differences regarding the number of alarms produced, and also differences in the logs produced. The results after running Metasploit showed some unexpected but clarifying results in the logs created. The whole process has been evaluated, and there has been given a summary of Snort, Bro and Suricata regarding installation, configuration and alarms.

Acknowledgments

I would like to thank my supervisor Hårek Haugerud, for being a huge resource and support through my thesis. Thank you for being helpful and understanding, and being available for needed meetings. I would also like to thank Amir M. Ahmed for providing me IP addresses and network for my test machine. And a special thanks to family, friends and classmates for keeping up my motivation and being understanding during this thesis.

Contents

Acknowledgments	1
1 Introduction	4
1.1 Motivation	4
1.2 Threats, security and IDS	6
1.2.1 Problem statement	7
2 Background	8
2.1 Computer threats	8
2.1.1 Malware	9
2.1.2 Purposes	9
2.2 Computer security	10
2.2.1 Intrusion detection system	12
2.2.2 Metasploit framework	17
2.3 Literature	18
3 Experimental setup and Methodology	20
3.1 Introduction	20
3.2 Test environment	21
3.3 Approach 1	22
3.4 Approach 2	23
4 Results	25
4.1 Approach 1	25
4.1.1 Snort	25
4.1.2 Bro	29
4.1.3 Suricata	36
4.2 Approach 2	43
4.2.1 Snort	44
4.2.2 Bro	48
4.2.3 Suricata	55
5 Analysis and Discussion	59
5.1 Introduction	59
5.2 Snort	60
5.2.1 Installation and configuration	60
5.2.2 Approach 1	60
5.2.3 Approach 2	61
5.3 Bro	62

CONTENTS

5.3.1	Installation and configuration	62
5.3.2	Approach 1	63
5.3.3	Approach 2	64
5.4	Suricata	64
5.4.1	Installation and configuration	64
5.4.2	Approach 1	64
5.4.3	Approach 2	65
5.5	Conclusion	66
5.5.1	Installation and configuration	66
5.5.2	Approach 1 and 2	66
6	Appendix	70
6.1	How to configure the HP Procurve 1800-24g switch	70
6.2	How to install Snort, Bro and Suricata	70
6.2.1	Needed packages and libraries	70
6.2.2	Installation	74
6.2.3	Snort	77
6.2.4	Bro	78
6.2.5	Suricata	79
6.3	How to run Snort, Bro, Suricata and tcpdump	81
6.3.1	Snort	81
6.3.2	Bro	81
6.3.3	Suricata	82
6.3.4	Tcpdump	82
6.4	How to install Metasploit framework	82

Chapter 1

Introduction

1.1 Motivation

The last few years the amount of malicious traffic on internet have increased enormously, [1] in a way that should be looked into with concern. Malicious traffic can cause loss or harm of data on computers, and loss of sensitive information is something that occasionally happens. [2]

Companies spend billions of dollars on computer security each year, [3] but still computers get infected or compromised by malicious traffic. Unaware employees, hackers, organized cyber criminals and unknown vulnerabilities are different reasons to why systems get compromised. Large companies and governmental organizations need the best tools available to prevent intrusions, since their companies are more exposed to malicious traffic.

The best firewalls, antivirus, intrusion prevention systems and other security tools available are expensive. Large companies can afford this tools, but what about smaller companies? They do not have the same economy, and not the same need of the best and most expensive computer security tools.

One common rule companies use about security, is that the amount of money spent on security should not be higher than the cost of loss of data or compromised computers. This is something companies should find out before investing in security tools. Normally large and governmental companies have more important information to protect than smaller companies have, and therefore they need to invest in the best security tools available. Smaller companies does not have the same need, and should they invest the same amount of money in security tools? In most cases they don't have the same need, and therefore is open source an area that should be interesting these kinds of companies.

The open source community have some advantages compared to payment solutions, where one of the advantages are the available source code. People using open source software can collaborate and contribute with own results and experiences to help each other to solve problems and for improvement.

1.1. MOTIVATION

When buying a firewall, an intrusion detection system or an intrusion prevention system, one often get a preconfigured box which are set up in the network and left there. By using open source security tools, one can interact with other open source users, search for solutions, and get much more information.

An intrusion detection system (IDS) is a well known security tool used by companies to prevent loss and harm of data. Companies such as Palo Alto and Sourcefire are two of the leading companies in this business. Palo Alto has only got a payment solution while Sourcefire has both a payment solution and a open source solution, the Snort IDS. In addition to Snort, there exist two other known open source intrusion detection systems, Bro and Suricata. Snort and Bro have existed since 1998, while Suricatas first stable version was released in July 2010.

An open source intrusion detection system is a good option for companies and organizations which do not have the same amount of money as the larger companies and governmental organizations have. When choosing open source intrusion detection system, one should have some knowledge about how to set them up, how to use them, and how to respond to the different alarms they create when running them in the network. It is difficult to know which intrusion detection system to choose without any previous knowledge.

Reality

Choices you make as leader of a company will affect the quality the network security of the company. Internet brings the world into your office, and with both opportunities and threats. Attackers automatically search for poorly secured system, and take control of these to use it in further attacks against others. There is no reason to think that a small company will be overlooked.

It is important to be aware of both internal as well as external threats. A survey made by Symantec in 2010[1], showed that about 50 per cent of all incidents are caused by employees. It could be an employee who cause accidents, or employees causing harm with the purpose of taking revenge or earn money. Both internal and external threats may have serious consequences.

The management of companies often see IT security as a cost, and not as a tool to prevent the system from being compromised, and they want it to be kept as low as possible. Becoming a victim of computer crime or accident can quickly become more expensive. How much do a business interruption, loss of working hours, cleanup work and lost sensitive information cost? Technical protection measures are important, but not worth anything if they do not work.

1.2 Threats, security and IDS

The amount of malicious traffic are increasing, and new threats are created every day. The threats are getting more and more serious, complex and sophisticated. There are no longer teenagers playing around creating viruses and worms that are the biggest problem for companies and organizations. Inside threats and organized cyber criminals looking for sensitive information, such as social security number and bank accounts, are some of the biggest problem today. These kinds of threats are getting worse, and companies need tools to prevent their system from being compromised.

When there are so many threats to be aware of, computer security becomes more and more important. The goal with computer security is to prevent property theft, corruption and natural disaster, and at the same time make sure that the information and property remain accessible for its intended users. As well, to protect valuable information and services from publication, tampering or collapse by unauthorized activities or untrustworthy individuals, and unplanned events.

Firewall is designed to deny or permit traffic based on preset rules. Even though if a firewall is well designed and configured, there exists threats that can pass through it. It could be malicious traffic that looks like normal traffic or cyber criminals hacking into the system. Most organizations find the need of additional hardware, software and network monitoring tools.

Antivirus is another computer security tool, which are designed to detect, prevent and remove malware. It is able to detect malware based on signatures and by anomaly detection. However, it is possible for a computer to be infected by new malware where there are no signature in the antivirus database. When a new malware is detected, countermeasures can be put in place to block or rid your computer of this type of code. But sometimes it can be too late and the harm is already done. It is desirable to stop malware in an earlier phase.

Firewall and antivirus used together gives a certain protection, but the question is if it is enough. Hackers can get passed a firewall, and computers can be infected and be compromised before the antivirus program detects it. Companies need some software and hardware that monitors the network for malicious traffic, and stops it before it does any harm. Intrusion detection system is a device or software application that monitors the network and/or system activities for malicious activities or policy violations and produces reports to system administrator.

There are two ways of setting up an intrusion detection system. One is host based intrusion detection, and the other is network intrusion detection. Network intrusion detection system is the most used. Snort, Bro and Suricata are three different network intrusion detection systems, and a network intrusion detection system monitors incoming, outgoing and internal traffic. When

1.2. THREATS, SECURITY AND IDS

malicious traffic is detected, alarms are created and sent as a report to network or system administrator.

1.2.1 Problem statement

When comparing Snort, Bro and Suricata there are different things one can compare, such as:

1. Compare the installation procedure
2. Compare alarms when run against normal traffic in a computer network
3. Compare alarms when run against known exploits from Metasploit

Chapter 2

Background

2.1 Computer threats

There are three separate but valuable components in a computer system; hardware, software and data. [4] By analyzing the security in a computer system, one can find how the system or the information can experience some loss or harm. An example is to find out which information one wants to protect, and make sure that it is only accessible for those who should have access.

A threat can be a set of circumstances that have the potential to cause loss or harm.[4] There exists many kinds of threats, including threats caused by humans and by computers. Unintentional human errors, design flaws and software failures are common errors in larger companies and organizations. Earthquake, flood and stolen equipments are threats as well, because it can damage the equipment or cause loss.

Computer systems can be compromised in many different ways; through crimeware, hacking, phishing, spam, social engineering, virus, worms and etc. Many cyber criminals look for known vulnerabilities in systems and utilizes this vulnerability to gain access to the system. There exists four kinds of threats: interception, interruption, modification and fabrication.[4] An interception is when an unauthorized persons, system or program has gained access to an asset. Illegal copying of program files or data files or wiretapping to obtain data in a network is examples of interception.

An interruption is when an asset becomes lost, unavailable or unusable. An example is malicious destruction of a hardware device or erasure of a program or data file. Modification is when an unauthorized person, program or system access and tamper with an asset. An example is if an unauthorized party change the value in a database, alter a program so that it perform different, or modify data being transmitted electronically.[4]

It is called a fabrication if somebody unauthorized party creates a counterfeit object which looks like the other objects on a system. The intruder can insert suspicious transactions to a network communication system, or add

2.1. COMPUTER THREATS

records to an existing database. Sometimes these additions can be detected as forgeries, but if skillfully done, they are almost impossible to distinguish from the real thing.

A malicious attacker must have three things to perform an attack; method, opportunity and motive.[4] The attacker must have the skills, knowledge and tools to pull the attack, the time and access to accomplish the attack, and a reason to want to perform this attack against this system. If one of this three are missing, the attacker will not be able to perform the attack. Anyway, is not easy to cut one of these off. [4]

It is difficult to determine an attacker's motive to attack a system. Some systems are attacked because they are attractive targets, which mean they are interesting to attackers. Other popular targets include law enforcement and defense department computers, because they are presumed to be well protected, and a successful attack will show skills.[4] And some targets are attacked just because they are easy targets or just because they are there; random, unassuming victims. Anyone can be a victim of an attack by an attacker with time, opportunity and knowledge.

2.1.1 Malware

Malware is short for malicious software, and it is software designed to cause harm or loss to a system without the owner knowing it.[5] It is an expression to mean a variety of forms of hostile, intrusive, or annoying software or program code. Malware includes viruses, worms, Trojan horses, dishonest adware, scareware, crimeware, most rootkits and other malicious and unwanted software.[5] The most common malware from criminals are by World Wide Web and email.

Software is considered to be malware based on the creators intention, rather than any particular feature. Malware is common by organized criminals, and the general lack of protection against newly produced malware, brings a new mindset for business on Internet, since some percentage of Internet customers will always be infected for some reason or another. And they need to continue doing business with infected customers.

2.1.2 Purposes

In the late 1990's and the start of 21th century, many viruses and worms were written as experiments and pranks. They were intended to be harmless or just annoying, and not cause any harm or loss. Sometimes the perpetrators did not realize how much harm the malware would do. Young programmers wrote them simply for practice, or to see how far they could go.

Since the broadband Internet access started to rise, malicious software has been designed for a profit. Since 2003, the majority of widespread viruses and worms were created to take control of users computers for black market

exploitation. Infected 'zombie' computers are used to send spam mail, to host contraband data such as child pornography, or to engage in distributed denial of service attacks as form of extortion.

Another malware that has emerged, is spyware, and it is designed to monitor users web browsing, display unwanted advertisements, or redirect marketing revenues to the spyware creator. Spyware do not spread like viruses; they are generally installed by exploiting vulnerabilities, or are packaged with user installed software.

2.2 Computer security

The objective of computer security is to prevent information and property theft, corruption, or natural disaster, while making the information and property remain accessible to its intended users. Computer security includes software, hardware, processes and mechanisms which intend to protect sensitive and valuable information from publication, tampering or collapse by unauthorized activities, or untrustworthy individuals and unplanned events. [6]

When talking about computer security, there are three important aspects of any related system; confidentiality, integrity and availability. Confidentiality makes sure that assets only get accessed by authorized parties, which means that those who should have access to something get that access. Access means reading, viewing, printing or knowing that particular asset exists.

Integrity means that assets can only be modified by authorized parties or only in authorized ways. Availability means that assets are accessible to authorized parties at appropriate times.[4] In other words, if some person or system has legitimate access to a particular set of objects, that asset should not be prevented. For this reason, availability is sometimes known by its opposite, denial of service.

One of the challenges when creating a secure system is finding the right balance among the goals, which often conflicts. For example, it is easy to preserve particular objects confidentiality simply by preventing everyone from reading that object. But by doing this, the files is not accessible for those who should have access, and it does not meet the requirement of availability. It must be a balance between confidentiality and availability.

There exist many different technologies and approaches to prevent the system from being compromised, or experience loss or harm. Firewall, antivirus, anti-spyware, intrusion prevention system, security policies, authentication, access control list, cryptography, backups and web scanner are tools that prevent computers from experience malware or malicious traffic.

Firewall

A firewall is the first line of defense for hosts connected to the Internet. It is a device that inspects all messages entering or leaving the intranet, and by looking at the content of the message it either accept or drop messages based on preset rules.[7] The standard way to set up a firewall is to deny all traffic, and then set rules that open for what kind of traffic you want to allow going through the firewall. If traffic does not match any of the preset rules, it will get stopped by the firewall.

There are several types of firewall techniques used to stop unwanted traffic, such as packet filters, application gateway, proxy server and network address translation (NAT).[8] They use different strategies or methods to stop unwanted traffic. Some of the methods the firewalls uses, is filter packets based on the information in the packets (combination of the packets source and destination address, its protocol, and, for tcp and udp traffic, the port number), understand certain applications such as FTP, DNS and HTTP, detect if an unwanted protocol try to get through a non-standard port or if a protocol is being abused in any harmful way.

Other methods firewalls use to stop unwanted traffic, is they can filter based on protocols, TTL values, netblock of originator, of the source, and many other attributes. It can inspect many different network flows, grant access to several networks and services, and it often cooperate with secure vpn system, intrusion detection systems, intrusion prevention systems, remote access service access (RAS) and other security systems

Even though firewall offer many methods to stop unwanted traffic, it has some weaknesses. The firewall is difficult to configure and keep up to date, and maintenance tasks become increasingly vulnerable for misconfigurations and errors.[9] And the firewall inspect only messages or packet passing through the firewall, and not traffic at the internal network. As well, clever hackers can pass a firewall and the firewall can create a form of an alarm, but it doesn't keep track of the intrusion or get information about it like an intrusion detection system would.

Antivirus

Antivirus is software used to prevent, detect and remove malware, such as computer viruses, computer worms, Trojan horses, spyware and other kinds of malware.[10] It uses different methods to these threats. The most common method is the signature based method, which detects computer viruses, computer worms and other kinds of malware by comparing the content of a file to a database or dictionary of malware signatures. But this method don't have the ability to detect new malware, since there doesn't exist a signature in the database yet.

2.2. COMPUTER SECURITY

To detect new malware, the anomaly based (also called heuristic) method is used. Many viruses' starts as single infection and through either mutation or refinements by other attacks can grow into many different variants.[10] Generic signatures is a anomaly based approach, and it can detect new viruses and variants of existing viruses by looking for known malicious code, or slight variations of such code, in files. These two methods, signature based and anomaly based, are the same methods as an intrusion detection system uses, but in little bit different way.

2.2.1 Intrusion detection system

An intrusion detection system can either be a device or a software application. Intrusion detection system focuses on detecting intrusions, logging information about them, and reports them to system administrator.[11] As well, organizations use intrusion detection system to identify problems with security policies, documenting existing threats and detect individuals that violate the security policy.

There are two main types of intrusion detection system (IDS); network IDS (NIDS) and host based IDS (HIDS). The NIDS is an independent platform that examines network traffic and monitoring multiple hosts to identify malicious activity. The usual way to set up a NIDS is by connecting it to a network hub, network switch configured for port mirroring, or network tap.[12] Sensors are placed at choke points in the network to be monitored, often in the demilitarized zone or at network borders. The sensors capture all network traffic and analyze the content of each packet for malicious content.

A host based IDS is placed on a host, where it detect intrusions by analyzing system calls, applications logs, file-system modifications, and other host activities and state. The sensors usually consist of a software agent.

There are two different techniques an IDS uses to detect malicious traffic/activity; statistical anomaly based IDS and signature based IDS. Statistical anomaly based IDS set up a baseline of what kind of traffic is seemed as normal. By looking at the bandwidth use, what protocols are used, what kind of ports and devices that are generally connected to each other, one can make a baseline of this, and when sample of traffic is outside this baseline, an alert will be created and send to the administrator.

The signature based IDS monitors the network and comparing the packets against preset signatures; which is signatures of known attacks. When the signature based IDS detect packets or traffic that match some of the signatures in the database, it will create an alarm about that malicious traffic and report to administrator. The problem with signature based IDS, is that it cannot detect malicious traffic where no signature has been made yet, but the statistical anomaly based IDS will be able to detect this.

2.2. COMPUTER SECURITY

There are some functions and concepts one should know of regarding intrusion detection system. An alert/alarm is a signal that suggests your system has been or being attacked, and this alarm can either be a 'true positive' or a 'false positive'. A true positive is if an intrusion has occurred and an alarm has been created, and a false positive is when the intrusion detection system thinks it is an intrusion and created an alarm, but the traffic was actually legitimate. 'False negative' is when an attack doesn't get detected, and 'false positive' is when no attack has taken place and no alarm has been created.

Site policy is guidelines within an organization that control the rules and configurations of an IDS, while site policy awareness is the ability an IDS has to dynamically change its rules and configurations in response to changing environmental activity.

Other functions or concepts one should know about, is confidence value, attacker or intruder, masquerader, misfeasor and clandestine user. Confidence value is a value of how effective the intrusion detection system detects or identify an attack.

An intruder or attacker, is a person, program or system that tries to gain unauthorized access to information, do harm or other form of malicious activity. Masquerader is an unauthorized user who tries to access the system as an authorized user, and a misfeasor is usually an internal user that either is an authorized user with limited permissions or a user with full permissions and who misuse their powers. A clandestine user is a user who acts as a supervisor and tries to use his privileges to avoid being captured.

Snort

Snort was created by Martin Roesch in 1998, and is an open source network intrusion detection and prevention system.

Snort can be used in three different ways; as a packet sniffer like tcpdump, a packet logger or as a network intrusion detection and prevention system.[13] When used as a packet sniffer, Snort will read network packets and display them on the console, and when used as packet logger Snort will log packets to disk. In intrusion detection mode it will monitor the network traffic and analyze the traffic against a rule set defined by the user.

In intrusion detection mode, Snort uses a number of rules that define anomalous traffic. Most of these rules are made by Sourcefire, and other rules are made by the community, and it is possible to make own rules as well. In addition to rules, Snort has several preprocessors which enable modules to view and alter packets before they get inspected by the intrusion detection system. [13]

When running Snort, it works by detecting and reporting malicious traffic or so called events. The process of reporting events can be configured

2.2. COMPUTER SECURITY

through event handling. By configuring thresholds one can reduce the number of logged alerts for noisy rules. This helps Snorts to handle more traffic. [2]

Snort has the capability to or can be configured to send output to various locations, when certain Snort rules is triggered. The most common output module is the alert syslog. Other output modules exist, such as 'alert fast' and 'alert full'. 'Alert fast' put a fast entry to the file specified, while 'alert full' sends the entire packet header along with the event message. [14]

Snort is capable of performing real time traffic analysis, which means that it can detect ongoing intrusions. It can perform logging on IP networks, perform protocol analysis, content searching and content matching, and it can be used to detect a variety of attacks and probes, such as bugger overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting and much more. [14]

Snort can be combined with other software, such as SnortSnarf, OSSIM, sguil, Snorby, Razorback and Basic Analysis and Security Engine (BASE) to provide visual representation of intrusion data. [13]

Another important thing with Snort is that one need to registrate at Snorts website to be able do download the ruleset. The official snort rules is the rules maintaiend by the vulnerability research team (VRT). Sourcefire has to keep Snort as an open platform, and they host rules submitted by the communiy (Snort users). These rules are distributed under the GPL and are freely available to all Snort users. [?]

Bro

Bro was founded by Vern Paxson in 1998, and is an open source UNIX based network intrusion detection system. Bro passively monitors network traffic and look for malicious traffic. It detect intrusions by first parsing network traffic and then execute event oriented analyzers that compare the activity with patterns deemed malicious.[15] The analyses include detection of specific attacks (signature and events) and unusual activities (anomalous).

Bro is normally placed at a key network junction, where it can be used to monitor all incoming and outgoing traffic. Bro provides functionality such as collecting, filtering and analyzing of network traffic. It is capable of giving a detailed analysis of popular protocols, and the output of this analysis is several events that describe the observed activity.

Bro comes with a set of policy scripts, which is designed to detect the most common internet attacks, while limiting the number of false positives, in example, alerts that confuse uninteresting activity with the important attack activity. Bro policy scripts are programs written in the Bro language, and the scripts contain rules that describe what kind of traffic or activities that are

2.2. COMPUTER SECURITY

looked as malicious. When analyzing the network activity, it initiates actions based on the analysis. [15]

The policy incorporates a signature matching facility that looks for traffic that matches these signatures. These signatures are expressed as regular expressions, and Bro's signature matching capability allows Bro to not only examine network content, but to understand the context of signature, greatly reducing the number of false positives. [15]

In addition to signatures, Bro can also analyze network protocols, connections, data amounts, incorporating it into analysis of new activity.

The policy script can generate output files of activity on the network, and it can generate problem alerts to event logs, including the operating system syslog. As well, the scripts can execute programs, which, in turn, send email messages, page the on-call staff, automatically terminate existing connections, or, with appropriate additional software, insert access control blocks into a routers access control list. [15]

A site can adapt Bro's operation by its specialized policy language and when new attacks is discovered. If anything is detected, Bro can generate a log entry, alert the operator in real time and execute an operating system command (terminate a connection for example). As well, Bro's detailed log files can be used to forensics.

Suricata

Suricata is an open source intrusion detection and prevention system developed by the 'Open Information Security Foundation'. The beta version was released in December 2009, while the first stable version came in July 2010. Suricata was created to bring new ideas and technology to the intrusion detection field. [16]

Open Information Security Foundation (OISF) provides Suricata with intrusion detection and prevention rule set, and the process of maintaining optimal security level is simplified by the Suricata engine. Suricata is able to use rules from different resources, such as Emerging Threats and Snort VRT rules, to provide the best rule set possible. [16]

As other network intrusion detection systems, Suricata monitor network traffic and create alarms/alerts logs when malicious traffic is detected. Suricata is designed to be compatible with other security components, and it offers features such as unified output functionality, and it is possible to accept calls from other applications through its pluggable libraries.

Suricata offers increased speed and efficiency in network traffic analysis with its multithreaded engine.[16] In addition to hardware acceleration, the

2.2. COMPUTER SECURITY

engine is build to utilize the increased processing power of the latest multi-core CPU. The engine supports and provides functionality such as the latest Snort VRT, Snort logging, rule language options, multithreading, hardware acceleration, unified output enabling interaction with external log management systems and IPv6.[17] As well, it supports and provides functionality such as rule based IP reputation, library plug ability for interaction with other applications, statistics output, and a simple and effective getting started user manual.

Sourcefire

Sourcefire Inc is a company founded in 2001 by Martin Roesch, and it develops network security and software, which is a commercial version of Snort software.[?] In addition, Sourcefire is committing to advancing open source technology and continue to maintain ties with the snort user community. [18]

Sourcefire is one of the leading companies within this field, and the company's initial growth was funded with 56.5 million dollar from several venture investors. Several companies have tried to buy the company without any luck, and at the end of 2008 the company had over 100 million dollar in cash and equivalents. In 2009 Sourcefire received the "Reader Trust" award for the best IDS/IPS solution for Snort, Network World's "2009 Best of tests" award for the Sourcefire 3D system. [18]

The Sourcefire 3D system is an intrusion prevention system solution that provides a layered security defense. The Sourcefire 3D modules include intrusion prevention system (IPS), real-time network awareness (RNA), real-time user awareness (RUA) and defense center (DC). [18]

Snort is an open source network intrusion detection and prevention system that uses a rule language which combines signatures and anomaly based inspection methods. The open source community has helped Snort getting developed to be the most widely used intrusion detection and prevention technology.

Palo Alto

Palo Alto Networks identifies applications independent of port, protocol, evasive tactics and SSL encryption. When applications identifies, they can be managed after the organizations policy. Many cleaver users use SSL for encryption to avoid proxy filters that exists in the network. Palo Alto Networks decrypts this SSL tunnel, so that the application that hides in the encrypted tunnel can identifies. [19]

Palo Alto Networks integrates with Active Directory or eDirectory to identify users. This means that the firewall rules can be made to apply exam rules, which give limited access to internet during exam. With good planning, access

2.2. COMPUTER SECURITY

can be managed by dragging groups into Active Directory and into specific groups.[19] With user information available, it is easy to identify machines that are attacked by virus, spyware and other threats.

Palo Alto Networks give unmatched overview of what happens in the network. With built in report generator for screen and PDF, the organization get a full overview of traffic in the network. Administrator get as well a control to let through or block specific traffic based on criteria's, such as zone, ip addresses, user/usergroup and application.[19] For example so can one usergroup, student, be accepted to use MSN, but not MSN file transfer. Access can be given for specific times during a day, for example youtube can be accepted in lunch and after school time, but not in the classes.

Palo Alto Networks can include both URL filtering and threat filtering. One uses Brightcloud, which is the most complete database for blocking chosen URL categories or simple URL's. Threat filtering filtrates threats such as virus, spyware, attacks on vulnerabilities, botnets and etc in real time.

Palo Alto Networks is not an UTM, but has a whole new architecture that perform all tasks in the firewall, with a central interface for administration. A policy gets created and includes all parameters, such as ip, user, application URL filter and threat filter. With a Palo Alto firewall, one can replace existing solutions for URL filters, IPS, inline virus filter and etc. In this way one can reduce costs for maintenance and annual maintenance costs.

Including to the functions described above, Palo Alto Networks includes as well functions like:

- Powerful routing fundament with RIP, OSPF and BGP
- Traffic shaping and policing
- Zone based architecture
- Virtualization
- Site-to-site IPSec VPN
- SSL VPN for remote access

2.2.2 Metasploit framework

The metasploit framework is a sub project of the open source Metasploit project, and the metasploit project is known for anti forensics and evasions tools, which is built into the framework.[20] The framework is a tool for developing and executing exploit code against a remote target computer.

Some of the most known exploits can be found in the metasploit framework. It can be used by security researchers to find potential vulnerabilities,

2.3. LITERATURE

but it can also be used by cyber criminals to break into systems.[21] When used by security researchers, vulnerabilities in systems can be found and fixed.

From the version 3.0, the metasploit framework have started to include fuzzing tools, which discover software vulnerabilities, rather than writing exploits for currently public bugs. The framework is run by first choosing and configuring an exploit, checking whether the intended target system is susceptible to the chosen exploit, choosing the encoding technique to encode the payload so that the intrusion prevention system (IPS) will not catch the encoded payload, and executing the exploit.[20] The possibility to combine any exploit with any payload is a major advantage, since it facilitates the tasks of attackers, exploit writers and payload writers.

When choosing an exploit and payload, one need information about the target system. Nmap is a tool that can get information about operating system version and installed network services, and nessus is a tool that can detect vulnerabilities.

2.3 Literature

Since intrusion detection system were invented in the middle of the 1980's, a lot of different intrusion intrusion detection systems have been developed, different methods to detect malicious traffic, different algorithms and other approaches as well. Intrusion detection systems use two different approaches to detect malicious traffic, which are signature based detection and anomaly detection.

Signature based detection is when the intrusion detection system uses a database with signatures of known malicious traffic, and compare these signatures against the traffic and see if there is some matches. If these signatures match any traffic on the network, alerts are created. Anomaly detection would detect statistical anomalies in the network traffic. The idea behind anomaly detection is to create a "baseline" that defines what kind of traffic that are deemed normal, while traffic that is outside this baseline are looked as malicious traffic and alerts are created.

The key advantage of signature detection is that signatures are easy to develop and understand if you know what network behavior you are trying to identify. The events generated by a signature based IDS can give you detailed information about what caused the alert. Signature based rules are based on pattern matching, and with modern day systems pattern matching can be performed very quickly. This is very important for multi-gigabit IDS systems. One can easily tweak signature based rulesets.

Since signature based IDS only can detect malicious traffic with known signatures, malicious with not known signatures will not be discovered. So

2.3. LITERATURE

called 0-day attacks cannot be detected by a signature based IDS. And the more signatures there are in the database, the slower will the detection engine be. As well, signature based IDS will create many false positives since they are usually based on regular expressions and string matching. Since they are based on pattern matching, they don't work well against different variants of the attacks as well.

Anomaly detection has the ability to detect 0-day attacks, if it falls out of the baseline that is set. It works very good against IRC based botnets and other malicious activity. It creates lower amount of false positives than the signature based IDS, and anomaly based IDS is very scalable, due to its architecture and method of operation. There is no need for creating new signatures for every attack and variant.

The disadvantage is that the anomaly detection engine is not able to decode and process the network protocols being analyzed in order to understand its goal and the payload. This is computationally expensive. In addition, there is very difficult to defined anomaly based rules, as every protocol analyzed by the system must be defined, base-lined and tested for precise thresholds. Most network protocols are implemented in a different way by different operating systems. As well, custom protocols needs to be analyzed, reverse engineered and require a lot of effort. Malicious activity which falls under normal usage pattern won't be detected by the anomaly engine.

Anomaly based IDS is the most researched method of these two. The signature based IDS are all about creating signatures of malicious activity, but anomaly detection has the strength of detect 0-day attacks, and all other malicious activity if the baseline is optimal. There is not so much one can improve by the signature based IDS, since is only uses signatures. While anomaly based IDS can be configured to stop all kinds of attacks, especially new malware. There are different methods within the signature based and anomaly based IDS that can be improved. The high amount of data on the high speed network, demands high packet processing.

Chapter 3

Experimental setup and Methodology

3.1 Introduction

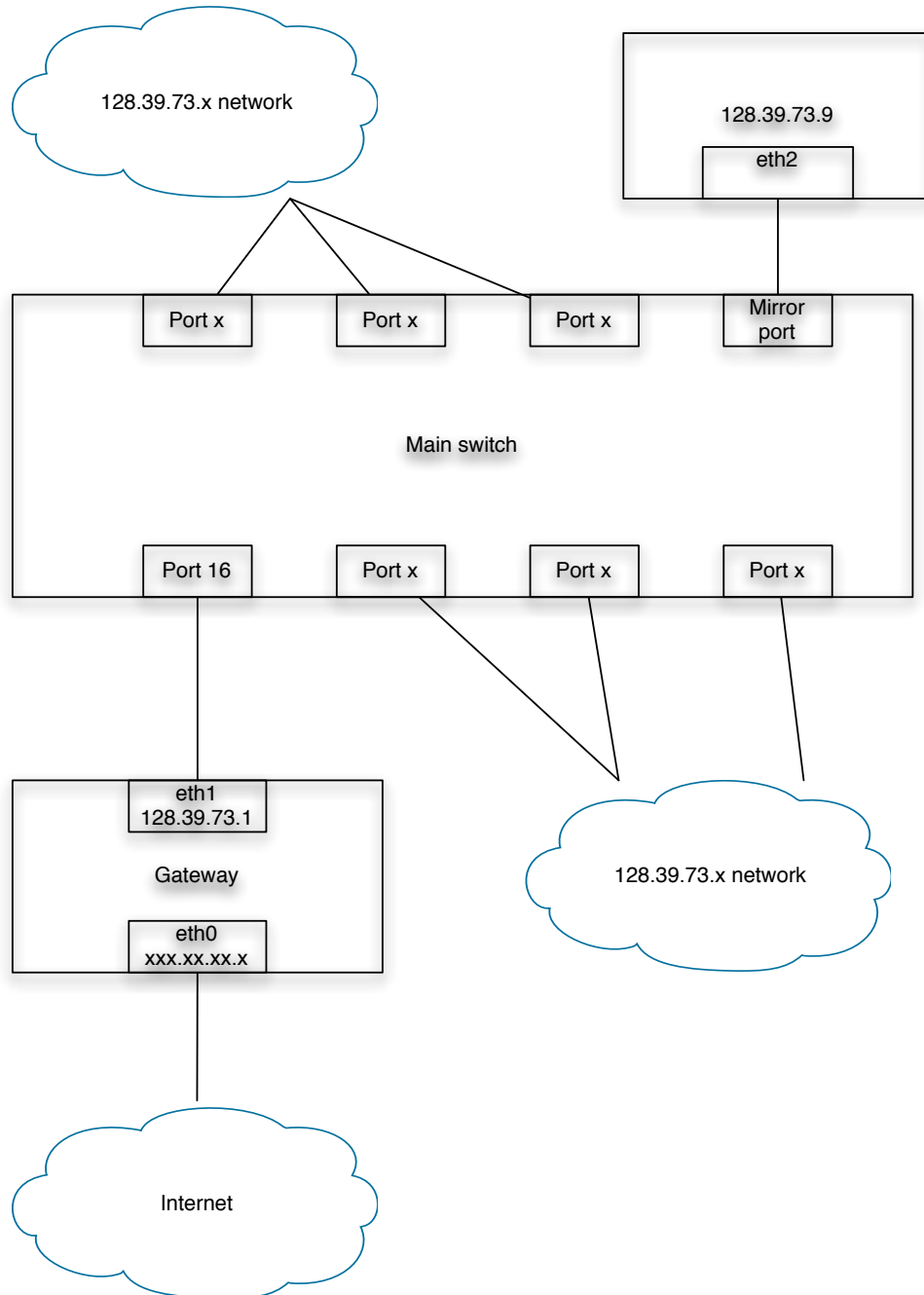
When comparing Snort, Bro and Suricata, one have to decide what kind of information there are possible to compare. There are many things that could be compared, such as output logs, alarms, configuration, ruleset, how to set them up, and test environment.

Snort and Suricata can be used with some of the same rules, but this is not done. Snort have been set to use the subscriber rules found at snort.org, while Suricata uses rules from Emerging Threats. Bro has their own ruleset with their own programming language.

Setting up Snort, Bro and Suricata were a time consuming process. Each of them required a set of installed packages, which helps them in the process of detecting malicious activity, and logging information about them.

The three intrusion deteciton systems were installed on the same machine, and not on different computers. The operating system were Debian Lenny, with a hard drive space of 66 GB.

3.2 Test environment



Picture are made by Petter Falch

3.3. APPROACH 1

This switch controls the 128.39.73.0/24 network. There are over 100 machines connected to this network, and when traffic are mirrored from the main port, which is port 16, to port 2 (mirror port on the picture), traffic from 100 machines are mirrored.

The machine where Snort, Bro and Suricata are installed, have the IP address '128.39.73.9' and the hostname 'jonas.vlab.iu.hio.no'. This machine is connected to port 2 on the switch, and receives traffic from this port.

Snort, Bro and Suricata were installed at jonas.vlab.iu.hio.no with the needed packages (see installation guide in appendix). The goal is to make Snort, Bro and Suricata analyze the same traffic, and compare them based on the logs created. Two approaches are described below.

3.3 Approach 1

After having installed Snort, Bro and Suricata at the machine jonas.vlab.iu.hio.no, and finding out how to run each of them, they were ready to run.

There are two ways of making the three intrusion detection systems analyze the same traffic. By running them at the same time against the interface that receives traffic from the switch, or by capturing traffic into a file by running tcpdump against the interface. Both methods were used.

1. Run Snort, Bro and Suricata against interface 2
How they are run are shown in appendix
2. Run tcpdump against interface 2
How tcpdump are run are show in the appendix

When the three intrusion detection create alarms and logs about the traffic either when run at the same time or against the tcpdump file, one can find out what kind of alarms that were triggered and compare these alarms. One could also compare other logs than the alarm logs, which Suricata and Bro creates.

One is often talking about 'true positives' and 'false positive' regarding intrusion detection systems. This is something that also can be compared. As well, the alarms created from Snort, Bro and Suricata differs, and there are no simple way to compare them.

- True positive: An intrusion or attack has occurred and an alarm have been triggered
- False positive: an alarm have been triggered, but the traffic is legitimate

3.4. APPROACH 2

A problem with intrusion detection systems are that they create to many false positives. This leads to more work for the system administrator to find the real threats, which needs to be stopped.

If this approach does not work, there is a backup plan where tcpdump is run and capture traffic over several days. If this approach is needed, the plan is to run it for four or five days, to get enough traffic to analyze. Each of the three intrusion detection systems will be run against this tcpdump file, and will produce logs and alarms out from this file. Tcdump will save the captured traffic in binary with this command:

3.4 Approach 2

Running Snort, Bro and Suricata at the same time over time, or running them against a tcpdump file that have capture traffic over some days, one will get a lot of alarms to analyze. Many of these alarms are probably false positives, which makes the analyzing part time consuming.

Another way to compare Snort, Bro and Suricata, is to test them against Metasploit Framework, which is exploits that can run be against the machine. The Metasploit Framework contains some of the most well known attacks, and by running different exploits against the machine where Snort, Bro and Suricata are installed, it will create alarms based on the exploits that is run. As the alarms are based on the traffic from the exploits, false positives will not occur because there are known attacks that is run. The Metasploit Framework were installed at a Ubuntu partition on the laptop.

Analyzing the traffic from the exploits sent by the Metasploit framework can be done in two ways. The first way is to connect the laptop directly to the machine where Snort, Bro and Suricata are installed, create a private network as there were problem with the wireless, and send the exploits towards the machine `jonas.vlab.iu.hio.no`.

Test 1 Metasploit:

```
jonas.vlab.iu.hio.no (128.39.73.9)
```

```
auto eth1
iface eth1 inet static
    address 10.0.0.3
    netmask 255.255.255.0
    network 10.0.0.0
    broadcast 10.0.0.255
    gateway 10.0.0.1
```

3.4. APPROACH 2

Laptop:

```
auto eth1
iface eth1 inet static
    address 10.0.0.2
    netmask 255.255.255.0
    network 10.0.0.0
    broadcast 10.0.0.255
    gateway 10.0.0.1
```

The second way to run the Metasploit Framework, is to run it through the network. By connecting a network cable to the laptop as it were problems with the wireless at the Ubuntu partition, and directing the traffic against the IP of jonas.vlab.iu.hio.no machine (128.39.73.9), the exploits were send towards the machine.

In both these ways, the traffic were captured by tcpdump into a file. The tcpdump file that captured the traffic when connecting directly to the machine ended with a size of 1.5 MB, while running tcpdump when the exploits were run three times through the network towards the machine, ended with a size of 14 MB. How the Metasploit Framework were run, are shown in the appendix.

Steps of how to run Metasploit framework:

1. Open ports on the target machine (jonas.vlab.iu.hio.no) with netcat.
2. Create a database in postgres (shown in appendix)
3. Entering the msfconsole and load the created database
4. Run nmap scan against 128.39.73.9 (jonas.vlab.iu.hio.no)
5. Information about the nmap scan gets saved in the database
6. Run metasploit with autopwn. (Shown in appendix)

At the same as starting running the exploits, tcpdump is set to capture traffic. After the exploits were finished running, tcpdump were stopped. Snort, Bro and Suricat were run against this tcpdump file.

Chapter 4

Results

4.1 Approach 1

The first approach was to run Snort, Bro and Suricata at the same time, in a way that they could analyze the same traffic. They were started at the same time and was set to run. After running for some time, it looked like they were not able to run simultaneously. Therefore tcpdump were used to capture traffic, and were set to run four days. The file ended with a size of 40 GB, and the three IDSs were run against this file.

Taking the time of how long each of them uses to analyze the captured traffic in the tcpdump file, is one way to compare the different intrusion detection engines. By typing 'time' in front of each command, the time would show after they finish running against the tcpdump file.

After running each of them against the tcpdump file, the results showed that:

- Bro used 27 minutes and 51 seconds
- Snort used 53 minutes and 19 seconds
- Suricata used 4 hours, 44 minutes and 37 seconds

Since Suricata were significantly slower, it was run against the tcpdump file again, to make sure that there was nothing wrong with the first test. The second result showed that Suricata used 4 hours, 47 minutes and 18 seconds.

4.1.1 Snort

Snort's output module can be configured in several ways. The default is logging in decoded ASCII format and full alerts. The other logs can be added with different options when running snort. There are seven alert modes available at the command line: full, fast, socket, syslog, console, cmg and none. Six of these modes are accessed with the -A command line.

4.1. APPROACH 1

The 'fast alert' mode writes the alert in simple format with a timestamp, alert message, source and destination IP's and ports. The full alert mode is the default mode and will be used automatically if you do not specify a mode. Unsonck mode sends alert to a UNIX socket that another program can listen on. -A none turns off alerting, -A console sends alert to the screen, and -A generates cmg style alerts. Even though Snort has the possibility to log in different ways, the full alert mode is the one that is used.

When running Snort against the tcpdump file with a size of 40 GB, it produced an ASCII file with an size of 128 MB, while the full alerts file were 124 MB.

Below is an example of some alerts created by Snort out from the tcpdump file:

```
[1:402:8] ICMP Destination Unreachable Port Unreachable
[Classification: Misc activity] [Priority: 3]
04/15-14:51:09.916383 128.39.73.188 -> 128.39.74.66
ICMP TTL:64 TOS:0xC0 ID:993 IpLen:20 DgmLen:102
Type:3 Code:3 DESTINATION UNREACHABLE: PORT UNREACHABLE
** ORIGINAL DATAGRAM DUMP:
128.39.74.66:53 -> 128.39.73.188:37617
UDP TTL:63 TOS:0x0 ID:31313 IpLen:20 DgmLen:74
Len: 46 Csum: 54021
(46 more bytes of original packet)
** END OF DUMP
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2005-0068]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2004-0790]

[1:15306:6] WEB-CLIENT Portable Executable binary file transfer
[Classification: Misc activity] [Priority: 3]
04/15-14:51:16.064884 158.38.122.10:80 -> 128.39.73.103:1033
TCP TTL:58 TOS:0x0 ID:52116 IpLen:20 DgmLen:1500 DF
***A*** Seq: 0xACD5E21 Ack: 0xF5214E13 Win: 0x1920 TcpLen: 20

[1:384:5] ICMP PING
[Classification: Misc activity] [Priority: 3]
04/15-14:51:30.225405 189.103.172.151 -> 128.39.73.79
ICMP TTL:107 TOS:0x20 ID:54713 IpLen:20 DgmLen:28
Type:8 Code:0 ID:1144 Seq:16225 ECHO

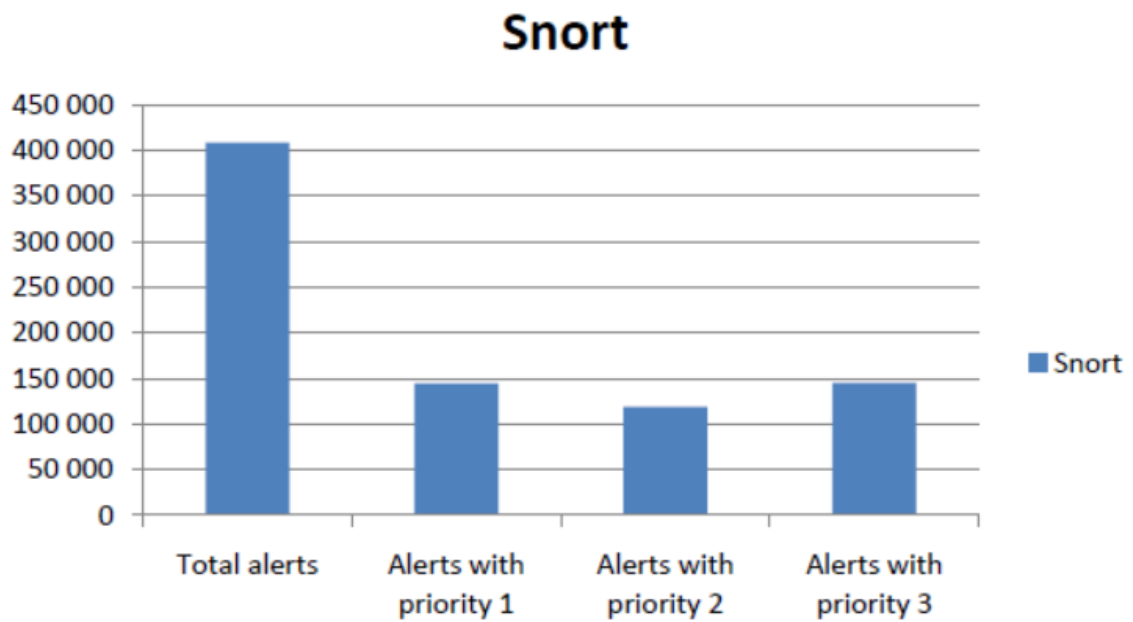
[119:19:1] (http_inspect) LONG HEADER
[Classification: Potentially Bad Traffic] [Priority: 2]
04/15-14:51:45.380352 128.39.73.51:51320 -> 65.55.57.252:80
TCP TTL:127 TOS:0x0 ID:1990 IpLen:20 DgmLen:1500 DF
***A*** Seq: 0xE8B6E47A Ack: 0x30ADE2F3 Win: 0xFAF0 TcpLen: 20
```

4.1. APPROACH 1

Information that can be found in each alarm in the full alert file, are:

- Snort ID
- Alarm message
- Classification
- Priority
- Timestamp
- Source and destination IP
- Source and destination port
- Protocols involved

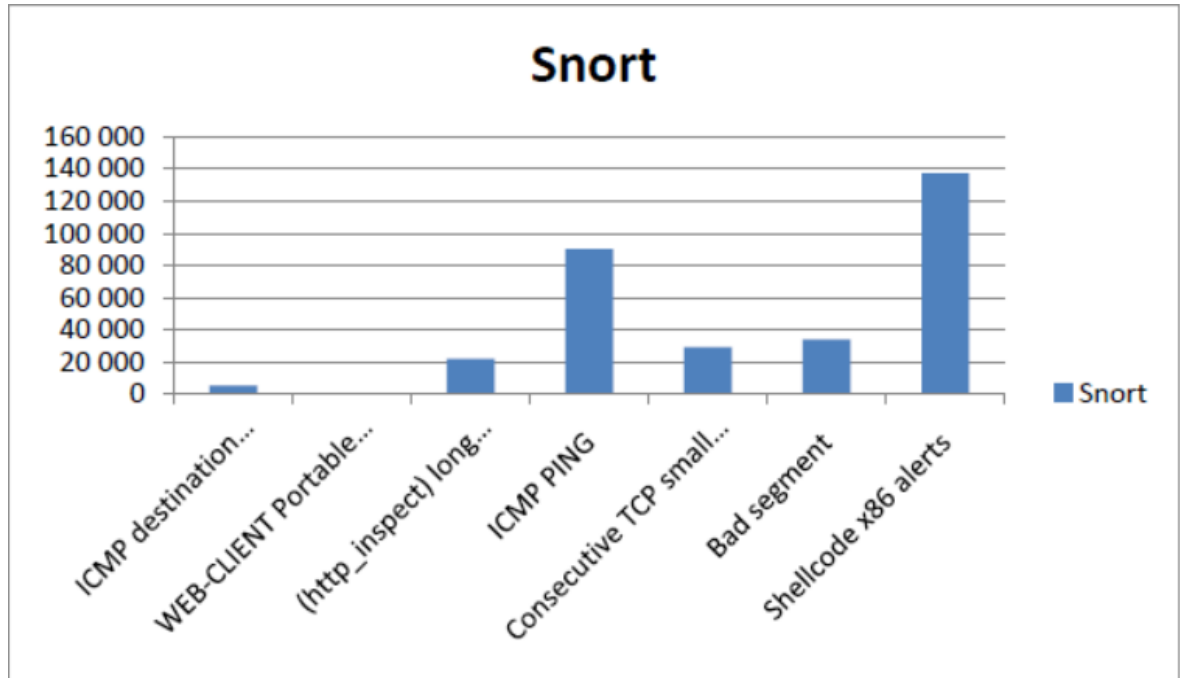
By performing some `cat`, `grep` and `wc -l` commands, one can find information about how many alarms that were produced, what kind of priority that appear most and how many times a specific alarm occurred.



- Total alerts: 408 390
- Alerts with priority 1: 144 475
- Alerts with priority 2: 118 855

4.1. APPROACH 1

- Alerts with priority 3: 145 062



Some of the alarms occurred quite often in the alert file. Two of the alarms were produced significantly more than the others, and that were 'Shellcode x86' and 'ICMP PING'. They occurred 137 156 and 90 029 times.

The 'Http inspect' alarm occurred 21 752 times in the alert file, the 'Consecutive TCP small segments' alarm occurred 29 170, and 'bad segment' were found 33 952 times. The two last alarms listed above, 'ICMP destination unreachable' and 'web-client portable executable' occurred respectively 5358 and 244 times.

What does the different alerts mean? By looking up the snort id [22], one can get a summary of the specific alert. Information about snort id, summary, impact, detailed information, affected systems, attack scenarios, ease of attack, false positive, false negative, corrective action, contributors and additional references can be found.

For example looking up the alert message 'ICMP Destination Unreachable Port Unreachable', the 'detailed information' about this alert tells that it is not an attack, but may indicate that the source of the packet was the target of a scan or other malicious activity. An ICMP Port Unreachable indicates that someone or something tried to connect to a port on a system that was not available.

4.1. APPROACH 1

By looking up the id to one of the 'Shellcode x86' alerts [22], for example 'SHELLCODE x86 inc ecx NOOP' with sid 1:269, it shows that this event is generated when an attempt is made to possibly overflow a buffer.

The alarm message 'WEB-CLIENT Portable Executable binary file transfer' indicates that a portable executable has been downloaded. And it can affect all Windows systems. The alarm message '(http inspect) long header' is generated when the http inspect preprocessor detects anomalous network traffic. In particular, the preprocessor has detected a long client header in an http request.

'ICMP PING' is generated when an generic ICMP echo request is made. An ICMP echo is used by the ping command to elicit an ICMP echo reply from a listening live host. This rule alerts on a generic ICMP request where no payload is included in the message or the payload not match more specific rules.

The 'Bad segment' are generated when the stream5 preprocessor detects anomalous network traffic. The preprocessor has detected a bad segment, the overlap adjusted size is less than or equal to 0 [22]. Information about the alarm 'Consecutive TCP small segments exceeding threshold' could not be found.

4.1.2 Bro

After running Bro against the tcpdump file, it produced 66 log files, where 12 of them were unique. 11 of them created 6 logs, while the alarm log were created five times, and there were one debug log. In addition to the alarm log and debug log, the 'conn log', 'ftp log', 'http log', 'irc log', 'notice log', 'signatures log', 'smtp log', 'software log', 'step log' and 'weird log' got created as well.

There were not all logs that contained information, even though they were created. Those who contained information after being run against the tcpdump file, were 'alarm log', 'conn log', 'ftp log', 'http log', 'notice log', 'smtp log' and 'weird log'. The most important or useful logs are the 'alarm log', 'conn log', 'notice log' and 'http log'.

The alerts in the alarm log looks like this:

```
t=1303084428.715139 no=PortScanSummary
na=NOTICE_ALARM_ALWAYS sa=212.174.139.12
num=31 msg=212.174.139.12\
scanned\ a\ total\ of\ 31\ ports tag=@13-2594-2679f

t=1303089397.301801 no=AddressScan
na=NOTICE_ALARM_ALWAYS sa=61.176.193.197
```

4.1. APPROACH 1

```
p=4899/tcp num=20 msg=61.176.193.197\ has\  
scanned\ 20\ hosts\ (4899/tcp) tag=@13-2594-269da
```

```
t=1303089397.316718 no=ShutdownThresh  
na=NOTICE_ALARM_ALWAYS sa=61.176.193.197  
p=4899/tcp msg=shutdown\ threshold\  
reached\ for\ 61.176.193.197 tag=@13-2594-269db
```

```
t=1303090308.368929 no=ScanSummary  
na=NOTICE_ALARM_ALWAYS sa=61.176.193.197  
num=183 msg=61.176.193.197\  
scanned\ a\ total\ of\ 183\ hosts tag=@13-2594-26a3e
```

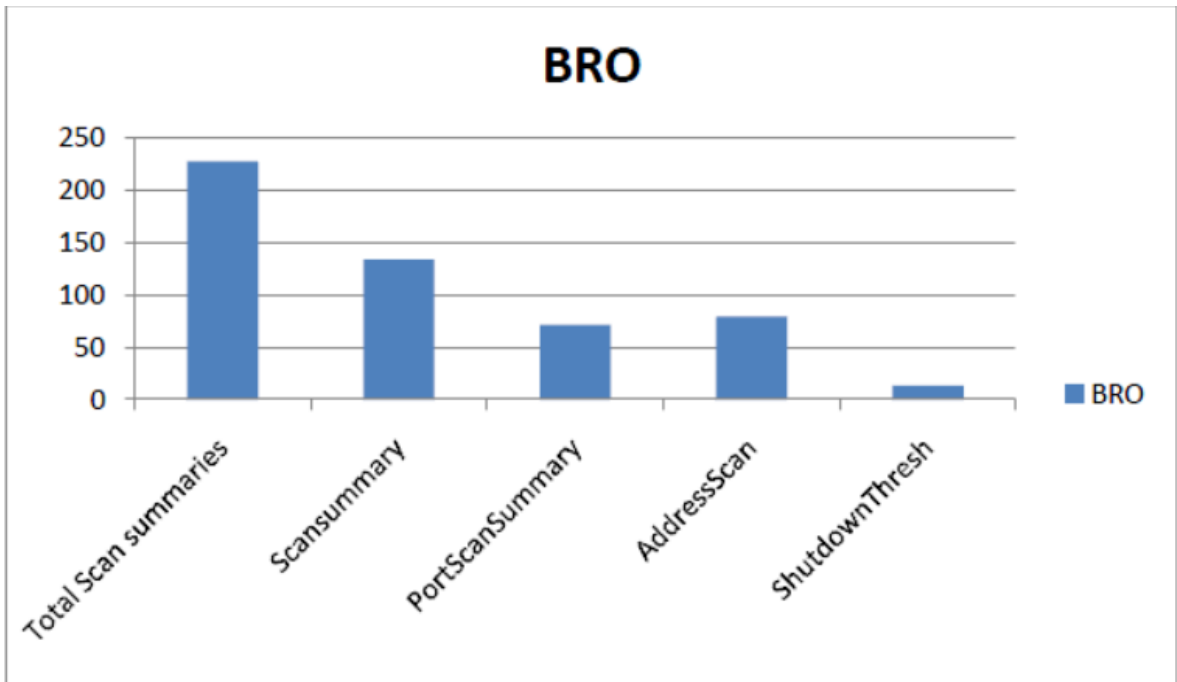
```
t: time in unix seconds  
no: notice  
na: notice action  
sa: source address  
sp: source port  
da: destination address  
dp: destination port  
msg: message  
tag: identifier
```

The first alert showed above, tells you that the time of the alert is Sunday 17 April 2011 23:53:48 GMT. The notice shows that the alert is a port scan from the IP 212.174.139.12, and there have been scanned a total of 31 ports. And by looking up this IP, one finds out that this IP is from Turkey.

Alarm log sums up the different alerts in a summary, rather than logging each alert many times. When some of the summaries shows that several ports or hosts have been scanned, it gathers this in one alert instead of printing many of the same alert.

The alarm log gives a summary of different scans, such as scan summary, portscan summary, address scan, and shutdown threshold.

4.1. APPROACH 1



ScanSummary tells how many hosts that have been scanned. Each of the ScanSummary contains information about how many hosts that has been scanned, and in one scan there could from 20 to 250 hosts scanned. The same applies to PortScanSummary, only how many ports scanned are shown in each summary. In each PortScanSummary there could be between 20 to 200 ports scanned.

AddressScan gives the same information as ScanSummary. It sums up how many hosts that have been scanned in each alarm created in the alarm log. The ShutdownThresh message tells how many times the connection have been terminated after a host have been scanned more than specified in the threshold.

The next log is the 'conn log', which shows all connections between hosts. The columns in this log are the time of the connection, the duration of the connection in seconds, local IP, remote IP, protocol, original bytes sent, response bytes sent, state, flags and tag. Some examples of the alarms are shown below.

Connection log:

```
1303077589.720744 7.042941 128.39.73.124
93.188.134.13 http 2934 80 tcp 500 301 RSTO X %299213

1303077588.625408 8.138393 128.39.73.124
195.88.55.16 http 2922 80 tcp 8741 49573 RSTO X %299201

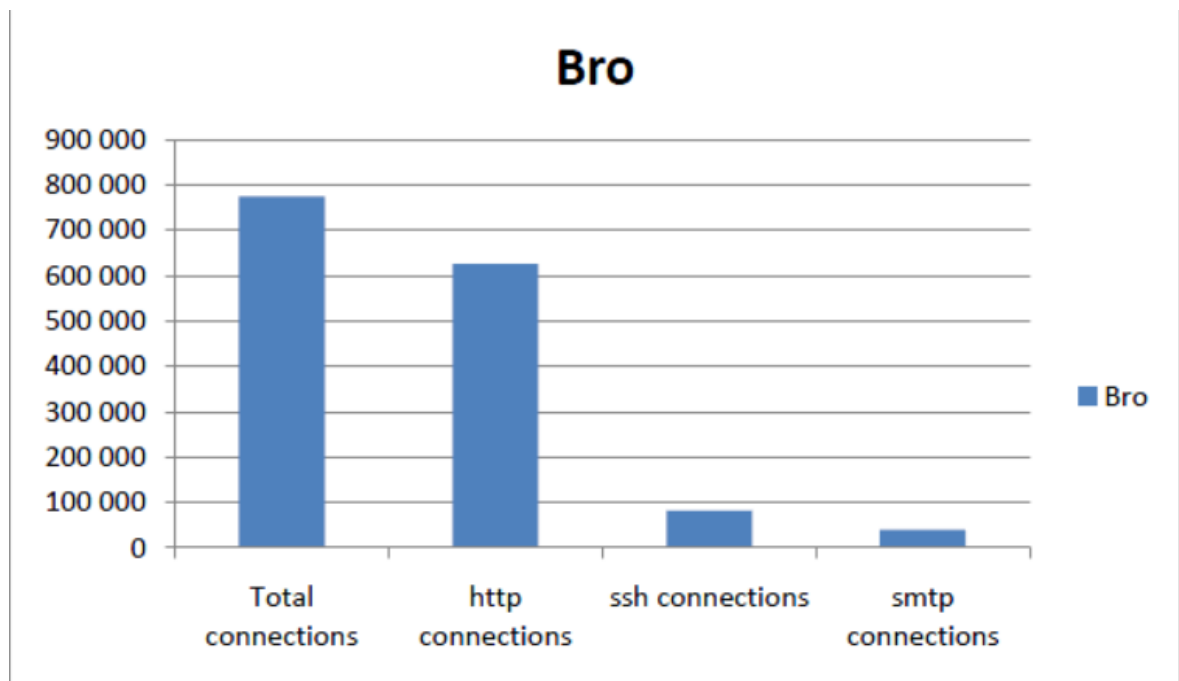
1303077588.865462 7.898387 128.39.73.124
```

4.1. APPROACH 1

```
195.88.55.16 http 2926 80 tcp 6887 3657 RSTO X %299205
```

The first connection above tells that the time of the connection was Sunday 17 April 2001 at 21:59:49 and the duration of the connection was about 7 seconds. The local IP is 128.39.73.124 and remote IP is 93.88.134.13, and the service is http. RSTO X means that a connection is established and the originator aborted.

Total connections, total http connections, ssh connections and smtp connections are shown in the table below.



Total connections were 775 545, and 625 929 of these were http connections. Total SSH and smtp connections were 81 328 and 39 352.

Notice log is the primary output facility in Bro. Information that can be found in the notice log, are:

- Timestamp
- Notice
- Notice action
- Source address
- Port and protocol used

4.1. APPROACH 1

- Message

```
t=1302872055.511079 no=AddressScan na=NOTICE_ALARM_ALWAYS  
sa=208.115.210.210 p=1068/tcp num=20 msg=208.115.210.210\  
has\ scanned\ 20\ hosts\ (1068/tcp) tag=@13-2594-15a
```

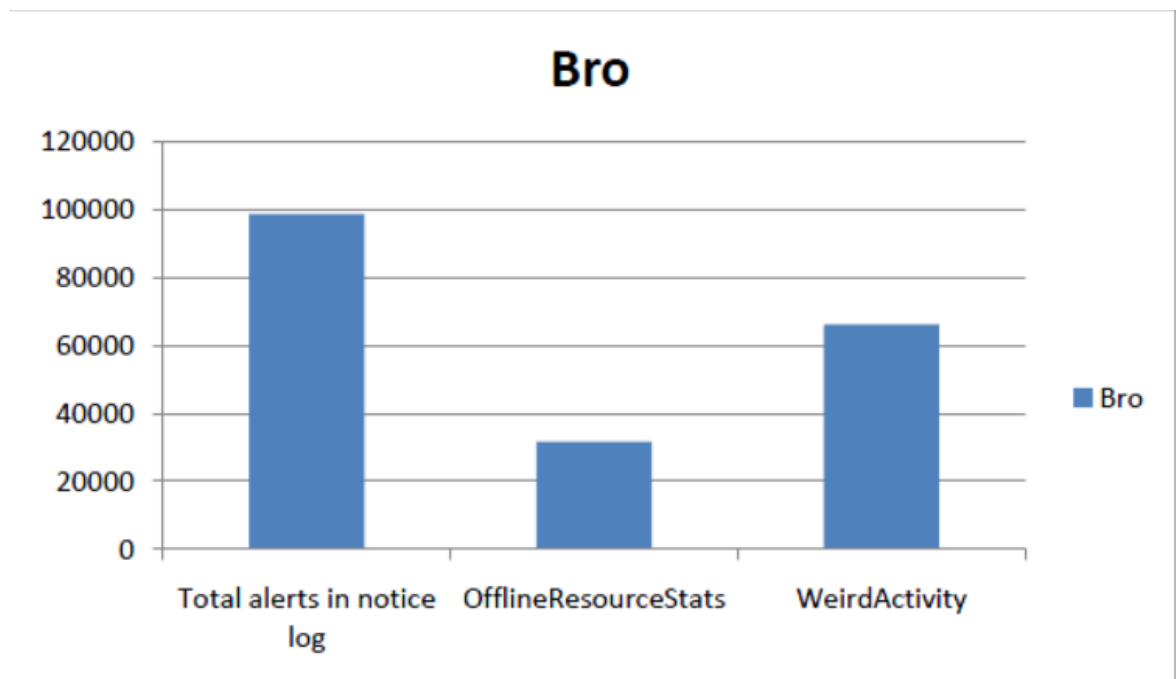
```
t=1302872098.424917 no=BackscatterSeen na=NOTICE_FILE  
sa=184.172.158.211 p=53/tcp msg=backscatter\ seen\  
from\ 184.172.158.211\ (20\ hosts;\ dns) tag=@13-2594-1a7
```

```
t=1302874127.196955 no=ScanSummary na=NOTICE_ALARM_ALWAYS  
sa=63.221.156.117 num=110 msg=63.221.156.117\ scanned\  
a\ total\ of\ 110\ hosts tag=@13-2594-5ee
```

The notice 'AddressScan' tells that several addresses at the network has been scanned from another machine, while BackscatterSeen is apparently flooding seen from source. ScanSummary is a summary of a scan activity.

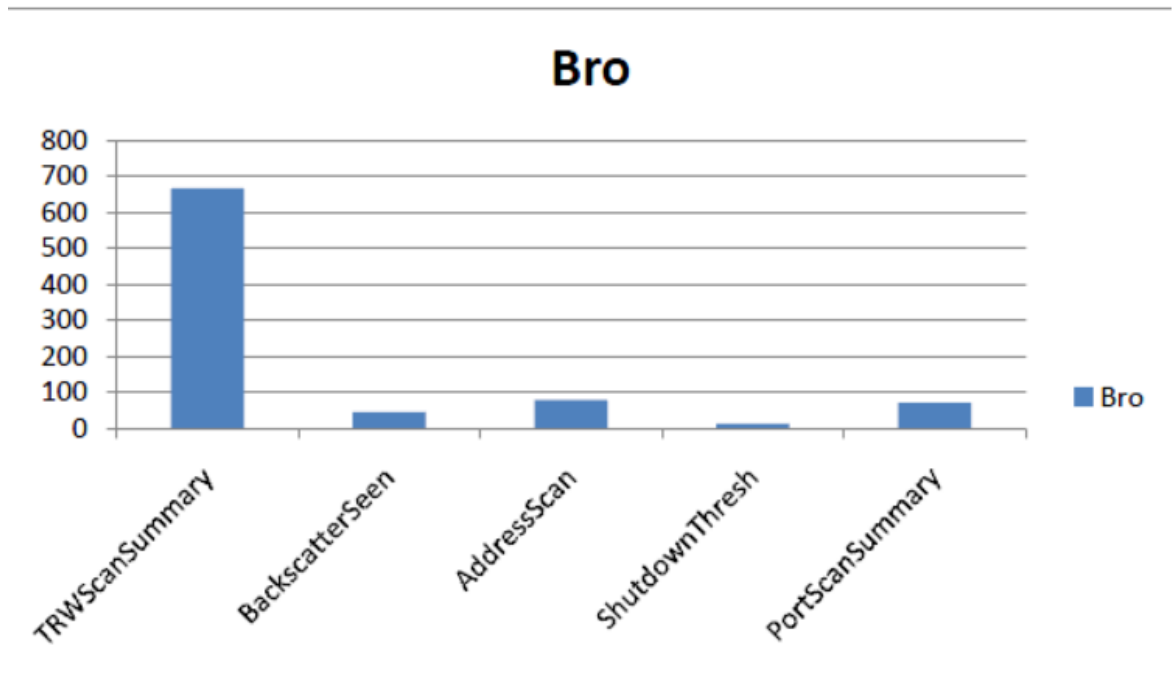
The first 'notice' from the notice log listed above, tells in UNIX time that the time were Friday 15 April 2001, 12:54:15, and the notice action was 'notice alarm always'. Source address were 208.115.210.210, source port 1068/tcp, and it tells that 20 host have been scanned.

The table below shows the total alerts in the notice log, and two of the alarm messages that occurred most.



4.1. APPROACH 1

The total amount of alerts in the notice log were 98 584, and 31 621 of them showed the message 'OfflineResourceStats' which tells how many events that is queued. 'WeirdActivity' message tells that there is packets with no IPv4 included. Rest of messages in the notice log are listed in the next table. There were 66 023 WeirdActiviy messages.



The message 'TRWScanSummary' occurred 667 times, and this message is a summary of scanning activities reported by TRW. 'BackScatterseen' is flooding attempts, and have been reported 45 times. 'AddressScan' and 'PortScanSummary' tells how many hosts and ports that has been scanned. AddressScan are listed 79 times, while PortScanSummary are listed 71 times in the notice log.

Each ScanSummary notice gives information about how many hosts that have been scanned, which usually are between 20-250 hosts.

The ftp analyzer generates summaries of ftp sessions; looks for sensitive usernames, access to sensitive files, and possible fpt bounce attacks, in which the host specified in a port does not correspond to the host sending the directive.

Here is an example of a ftp session:

```
1303142310.223106 #3 128.39.73.55/12613 > 217.69.76.55/ftp start
```

4.1. APPROACH 1

```
1303142310.338454 #3 response (220 Service ready for new user.)
1303142310.338913 #3 USER anonymous/-wget@ (logged in)
1303142310.416644 #3 SYST (215 UNIX)
1303142310.455136 #3 PWD (done)
1303142310.493927 #3 TYPE I (ok)
1303142310.532418 #3 CWD /gcrypt (ok)
1303142310.610500 #3 SIZE libgcrypt-1.4.2.tar.gz (unavail)
1303142310.648893 #3 PASV (227 217.69.76.55/40199)
1303142310.729969 #3 RETR libgcrypt-1.4.2.tar.gz (unavail)
1303142316.342856 #3 finish
```

These ftp sessionss tarts with a connection where the IP 128.39.73.55 on port 12 613 connects to a a ftp server at IP 217.69.76.55. The ftp server response and tell that 220 services are ready for the new user. Then anonymous user log in to the system with password and the login is encrypted. Then the user try to get libgcrypt-1.4.2.tar.gz but this package is unavailable. The user try one more time, but still unavailable and the session finish.

The http log file shows the details from http streams. It contains a packet epoch time and stream reference value for each entry. Lines contain stream header summary information and content detail. This includes source and destination IP's, hostnames, server type, dates, content information, and html code.

HTTP log:

```
1303077614.669620 %299222 start
128.39.73.128:39256 > 209.85.148.106:80
```

```
1303077614.869910 %299222
GET /search?q=poker
(200 "OK" [49680] www.google.no)
```

```
1303077614.938078 %299223 start
128.39.73.128:57853 > 209.85.148.105:80
```

```
1303077615.085749 %299223
GET /search?q=strip+poker
(200 "OK" [42996] www.google.no)
```

```
1303077615.676208 %299226 start
128.39.73.128:54165 > 209.85.148.99:80
```

4.1. APPROACH 1

```
1303077615.877380 %299226
GET /search?q=beer+fong (200 "OK" [47954] www.google.no)
```

Each http request have a timestamp, and then the source IP and port, and destination IP and port. The next line tells what the user search for, which in the last http request above, are beer fong.

There were quite little information logged about smtp traffic. All smtp traffic that were logged were:

SMTP log:

```
1303113966.698958 #1 128.39.73.30/45472
> 158.36.161.27/587 start external
1303113966.708262 #1 STARTTLS
```

Weird log were the last log containing any information about the traffic captured in the tcpdump file. The weird log contains unusual events based on the weird policy file. It represents packets that Bro consider suspicious for miscellaneous reasons and should be investigated further.

Weird log:

```
1303077716.272441 66.249.72.68/50596 >
128.39.73.11/https: above_hole_data_without_any_acks

1303077905.914390 128.39.73.61/53145
> 213.150.61.61/http: unescaped_special_URI_char

1303079121.306419 128.39.73.98/49332
> 81.93.163.115/http: SYN_after_reset

1303078930.607053 124.232.152.99/4170
> 128.39.73.60/327: connection_originator_SYN_ack
```

4.1.3 Suricata

In the Suricata configuration file 'suricata.yaml' one can set Suricata to log into different logs. The first log specified is the 'fast.log', which is a line based alerts log similar to Snorts 'fast.log'. Suricata can as well be set to log in a way that it can be used by a program called Barnyard later, but this is not enabled. The other logs that Suricata creates are the 'http.log' which is a line based log of

4.1. APPROACH 1

http requests. The 'alert-debug.log' is a full alert log containing information about suspicious activity.

After running Suricata against the tcpdump file, it created four different logs. These logs are alert-debug.log, fast.log, http.log and stats.log. The total size of the alert-debug log was 40 MB, fast log was 11 MB, http log was 117 MB, and the stats log was 4 MB.

Information about the alarms created can be found in the alert-debug log and fast log. The alert-debug log describes every alarm very specific, while the fast log shows only the most important information about each alarm, and gives a better overview of each alarm. Since they contain information about the same alarms, results from fast log are shown since these alarms were simpler to read and understand.

Useful information that can be found in the fast log, are timestamp, sid number which tells which rule that has been triggered, alarm message, classification of the alarm, priority. IPs and ports involved, and two links were one can find information about the alarm and which rule that triggered the alarm. Three alarms from the fast log are shown below.

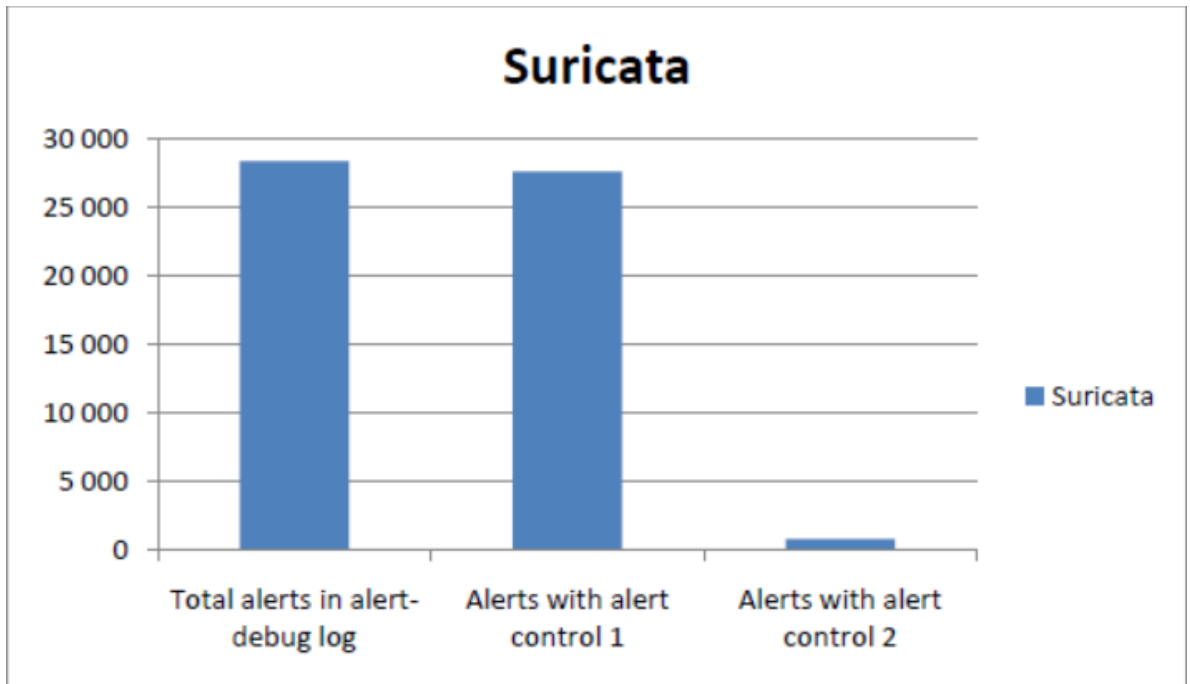
Fast log:

```
04/15/11-12:58:06.418709  [**] [1:2012204:3]
ET SCAN Modified Sipvicious Sundayddr Scanner [**]
[Classification: Attempted Information Leak] [Priority: 3]
{17} 202.109.115.169:5060 -> 128.39.73.71:5060
[Xref => http://code.google.com/p/sipvicious/]
[Xref => http://blog.sipvicious.org/]
[Xref => http://honeynet.org.au/?q=sunday_scanner]

04/15/11-13:13:33.496162  [**] [1:2001219:18]
ET SCAN Potential SSH Scan [**]
[Classification: Attempted Information Leak] [Priority: 3]
{6} 63.221.156.117:40269 -> 128.39.73.117:22
[Xref => http://en.wikipedia.org/wiki/Brute_force_attack]
[Xref => http://doc.emergingthreats.net/2001219]
[Xref => http://www.emergingthreats.net/cgi-bin/
cvswb.cgi/sigs/SCAN/SCAN_SSH_Brute_Force]

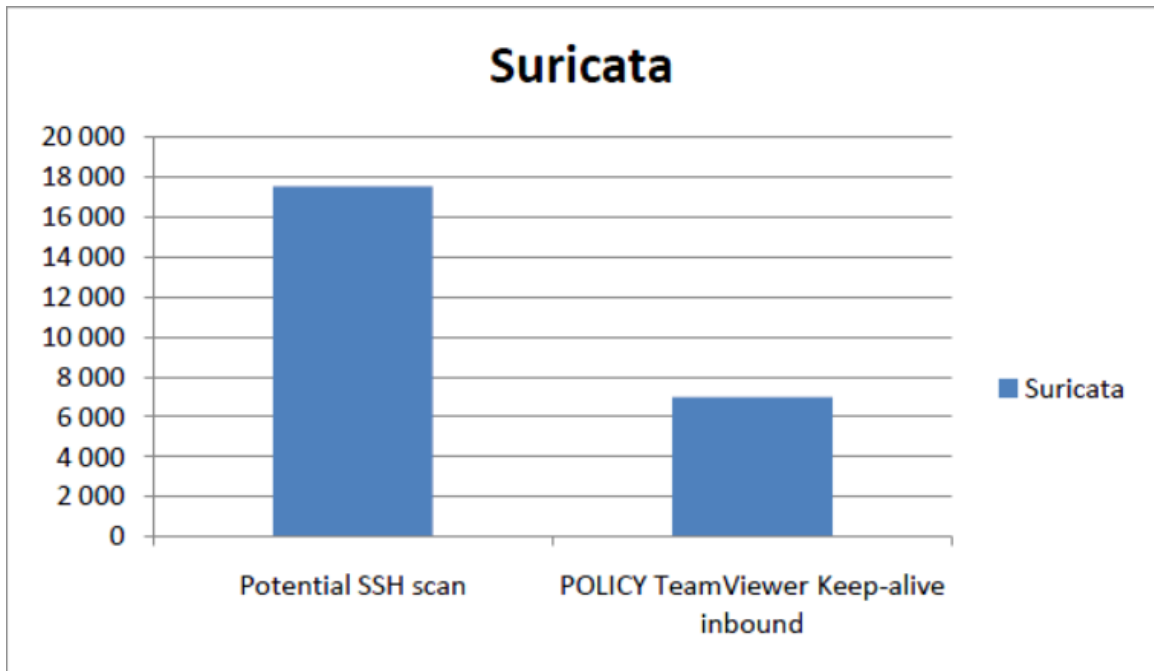
04/15/11-13:18:31.160912  [**] [1:2001331:8]
ET POLICY RDP disconnect request [**]
[Classification: Misc activity] [Priority: 3]
{6} 128.39.89.9:35014 -> 128.39.73.71:3389
[Xref => http://doc.emergingthreats.net/2001331]
[Xref => http://www.emergingthreats.net/cgi-bin/
cvswb.cgi/sigs/POLICY/POLICY_RDP_Connections]
```

4.1. APPROACH 1



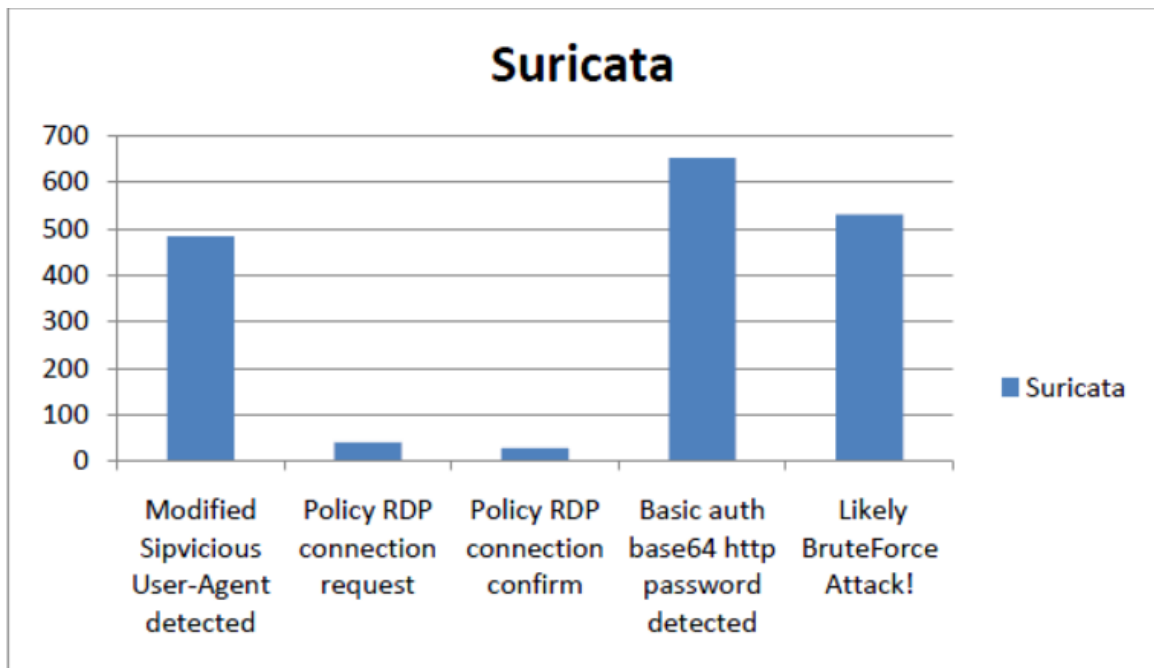
After reading the tcpdump file, Suricata created a total of 28 243 alarms in the alert-debug log. 27 594 of them had alert control 1, while 749 of them with alert control 2.

4.1. APPROACH 1

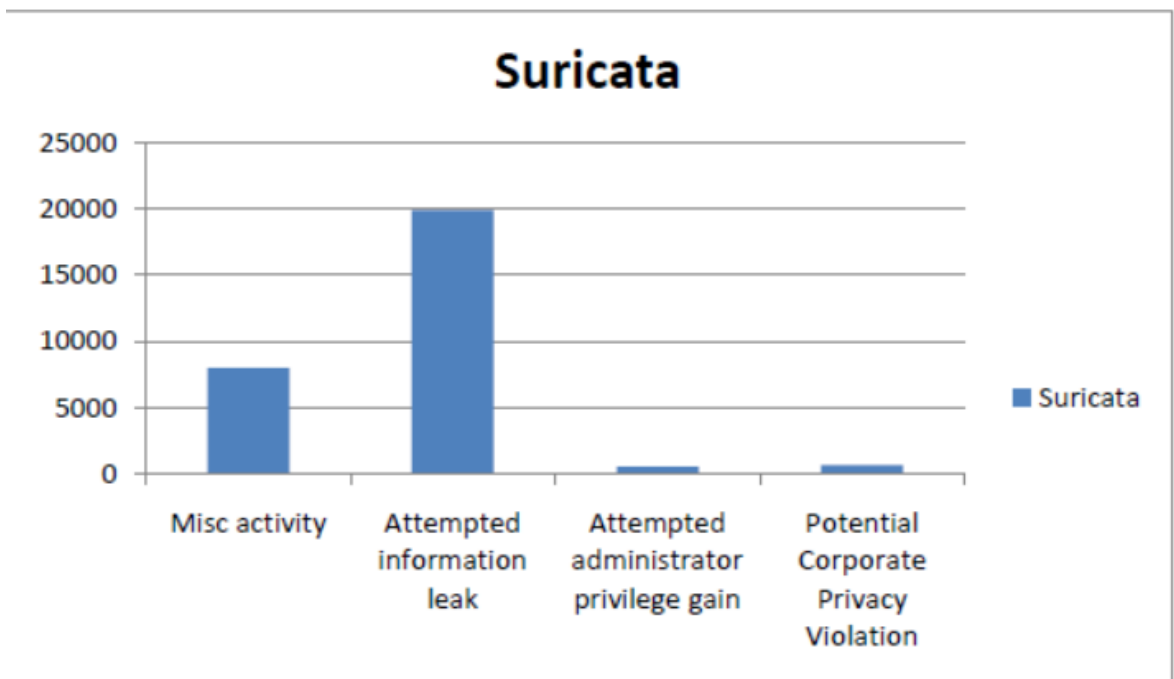


In the fast log, there is created as many as 17 499 alarms that says 'Potential SSH Scan', and there are 6986 alarms which says 'Policy TeamViewer keep-alive inbound'. The 'Policy TeamViewer keep-alive inbound' message is created because of a typo. The message in sid 2008795 which creates this alarm has a typo: 'TeamVieweer' should be 'TeamViewer'. SSH scan can be seen as a brute force attempt, where many combinations of username and password are tried to gain access.

4.1. APPROACH 1



The rest of the alarms in the fast log is listed in this graph. There are 483 alarms of 'Modified Sipvicious User-Agent detected', 651 'Basic auth base64' alarms, 530 'Likely BruteForce attack' alarms, 41 'Policy RDP connection request' and 29 'Policy RDP connection confirm' alarms.



4.1. APPROACH 1

Each alarm have a classification which tells what kind of category the alarm belongs to. 19 991 alarms belonged to the classification 'Attempted information leak', while 7980 alarms belonged to the classification 'Misc activity'. 652 of the alarms were potential corporate privacy violation, and it was 531 attempts to gain administrator privilege.

The http log contain information about each http request, similar to Bro, but contain some more information and are a bit more difficult to read. Each http request shows a timestamp, which address that was requested, message, and involved IP's and ports. Examples are shown below.

```
04/15/11-12:51:02.068859 security.debian.org [**]  
/dists/lenny/updates/Release [**]  
Debian APT-HTTP/1.3 (0.7.20.2) [**]  
128.39.73.49:80 -> 195.20.242.89:50520
```

```
04/15/11-12:51:02.820175 download.windowsupdate.com [**]  
/v9/windowsupdate/a/selfupdate/WSUS3/x86/  
Other/wsus3setup.cab?1104151251 [**]  
Windows-Update-Agent [**]  
128.39.73.103:1028 -> 65.54.89.112:80
```

```
04/15/11-12:53:12.822062 128.39.73.231 [**]  
/sdk/vimService?wsdl [**] <useragent unknown> [**]  
128.39.28.174:80 -> 128.39.73.231:49632
```

The http log is created to show details from http streams. After some there are most malicious traffic on the web, there is useful to have the http log which saves information about each http request. As well, the http requests can be compared against the alarm found in the alert-debug log and fast log.

Stats log is the last log Suricata created. This log gather information about how many packets, how many bytes, how many IPv4 and IPv6, how many tcp and udp packets there are each eight second. Other information such as icmp packets and average and max package size can be found as well. Below there is show an example of the stats.log

4.1. APPROACH 1

Stats log:

26/4/2011 -- 22:21:00

Counter	TM Name	Value
decoder.pkts	Decode & Stream	15314
decoder.bytes	Decode & Stream	13947880
decoder.ipv4	Decode & Stream	15254
decoder.ipv6	Decode & Stream	2
decoder.ethernet	Decode & Stream	15314
decoder.raw	Decode & Stream	0
decoder.sll	Decode & Stream	0
decoder.tcp	Decode & Stream	15079
decoder.udp	Decode & Stream	163
decoder.icmpv4	Decode & Stream	5
decoder.icmpv6	Decode & Stream	0
decoder.ppp	Decode & Stream	0
decoder.pppoe	Decode & Stream	0
decoder.gre	Decode & Stream	0
decoder.vlan	Decode & Stream	0
decoder.avg_pkt_size	Decode & Stream	910.792739
decoder.max_pkt_size	Decode & Stream	1514
defrag.ipv4.fragments	Decode & Stream	4
defrag.ipv4.reassembled	Decode & Stream	2
defrag.ipv4.timeouts	Decode & Stream	0
defrag.ipv6.fragments	Decode & Stream	0
defrag.ipv6.reassembled	Decode & Stream	0
defrag.ipv6.timeouts	Decode & Stream	0
tcp.sessions	Decode & Stream	86
tcp.ssn_memcap_drop	Decode & Stream	0
detect.alert	Detect	0

4.2 Approach 2

The second approach was to run Metasploit against the machine where Snort, Bro and Suricata were installed. After Metasploit had been installed, there were two ways that Metasploit could be run. One could run Metasploit by connecting directly to the machine by a twinned cable and create a private network, or one could run Metasploit through the network. Metasploit were run both ways.

```
jonas.vlab.iu.hio.no (128.39.73.9)
```

```
auto eth1
iface eth1 inet static
    address 10.0.0.3
    netmask 255.255.255.0
    network 10.0.0.0
    broadcast 10.0.0.255
    gateway 10.0.0.1
```

Laptop:

```
auto eth1
iface eth1 inet static
    address 10.0.0.2
    netmask 255.255.255.0
    network 10.0.0.0
    broadcast 10.0.0.255
    gateway 10.0.0.1
```

After running a nmap scan against the target machine, it gathered information about which ports that were open, and saved this information to the database that were created. By having this information, the Metasploit framework could use another database containing exploits for different ports, to send these exploits against those ports saved in the database from the nmap scan.

To be sure that Snort, Bro and Suricata got the same traffic to analyze, tcpdump were run to capture the traffic from the Metasploit exploits sent towards the target machine. The first test were the laptop were connected directly to the target machine, tcpdump captured the traffic into a file, with ended with a size of 1,5 MB after Metasploit exploits were finished running.

The second test where Metasploits were run through the network, the Metasploit exploits were run three times to create a tcpdump file, and make Snort,

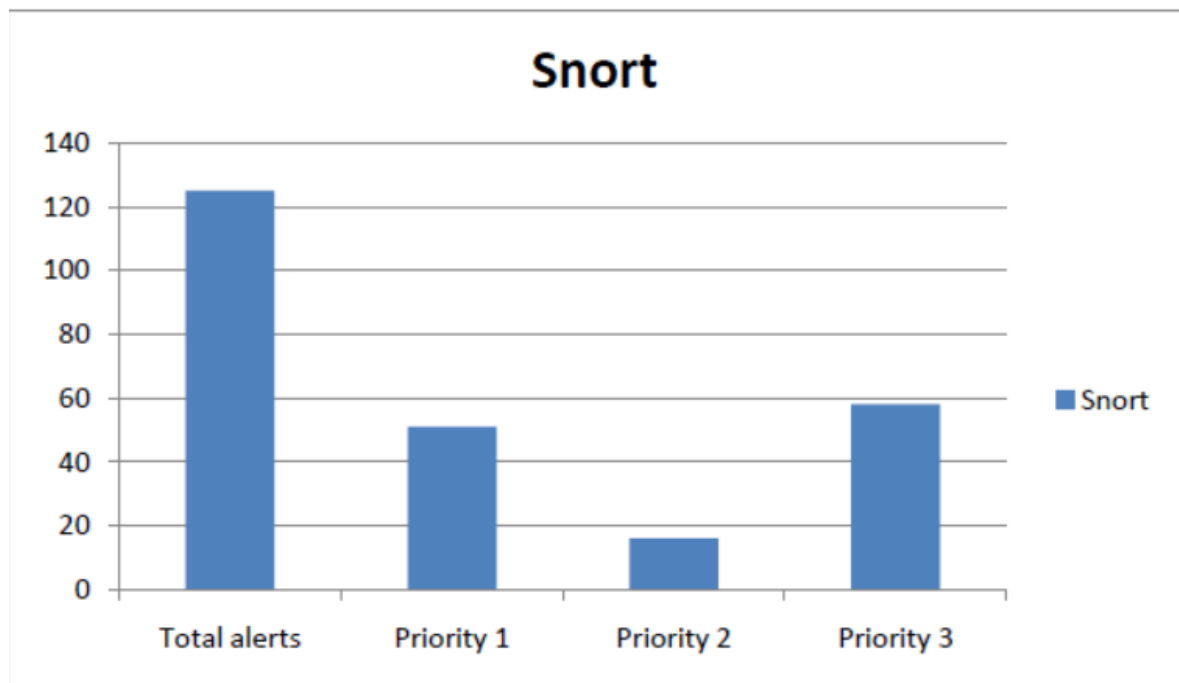
4.2. APPROACH 2

Bro and Suricata to analyze more traffic than in the first test. The tcpdump file ended with a size of 14 MB.

4.2.1 Snort

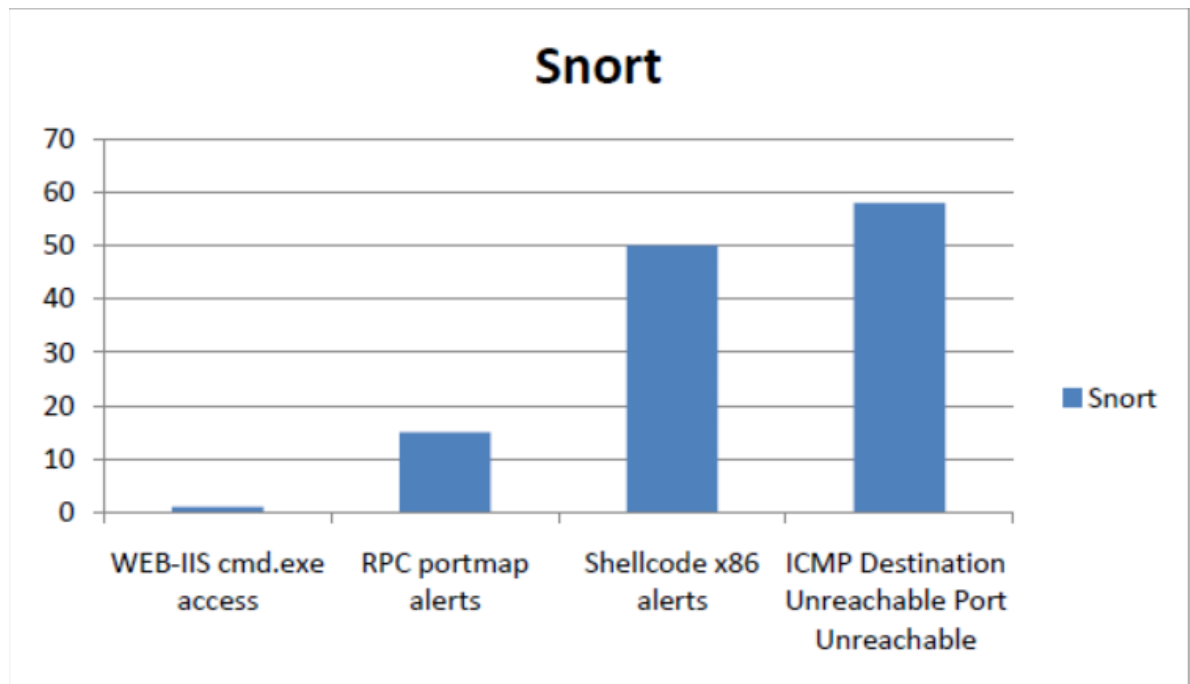
As in approach 1, Snort were set to log into the default ASCII and full alerts file. But the results below are based on the full alerts file.

After going through the results from the first and second test with Metasploit, it showed that some of the same alarms were triggered as in approach 1, and there were some new alarms that did not appear in approach 1.



The results above shows the the total amount of alarms created after running Snort against the first tcpdump file that captured traffic from the Metasploit attack. 125 alarms were created, 51 of them with priority 1, 16 with priority 2, and 58 with priority 3.

4.2. APPROACH 2



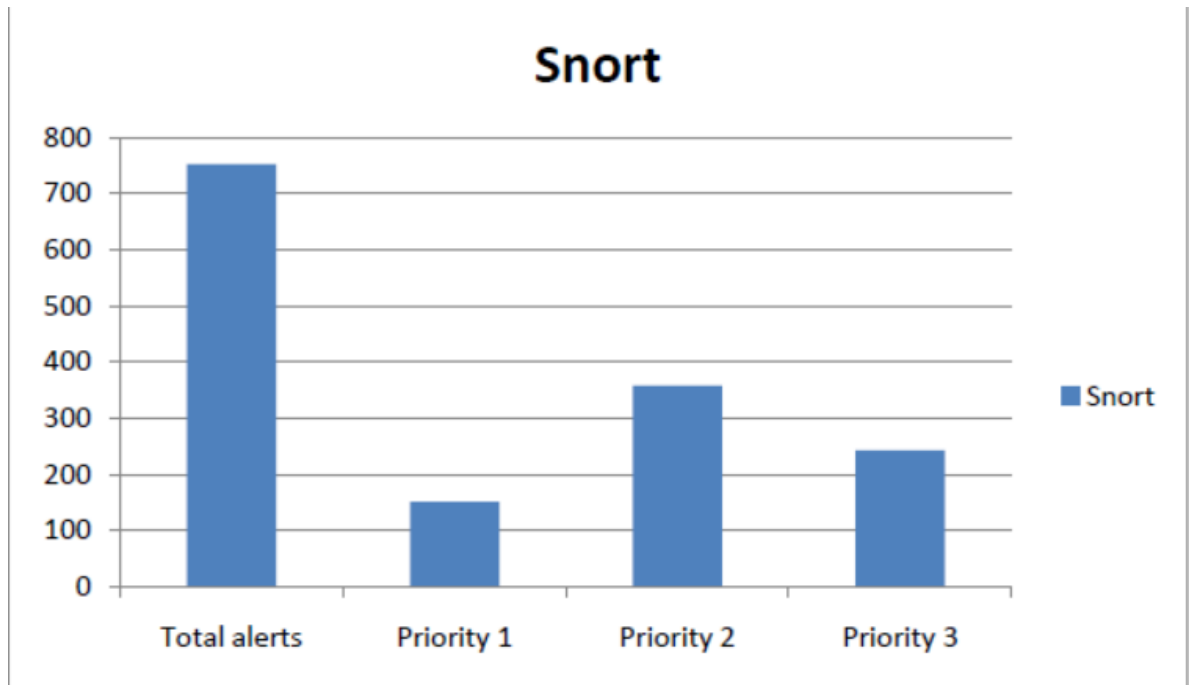
This graph tells how many times each different alarm appeared in the full alert file. There were only one 'WEB-IIS cmd.exe access' alarm, any by looking up at the snort id [22], one can find out that this event indicates an attempt to exploit potential weaknesses in a host running Microsoft IIS.

There were 15 different 'RPC portmap' alarms. The RPC message 'RPC portmap Solaris sadmin port query udp request' is an event generated when an attempt is made to exploit known vulnerability in Solaris. This attack can cause 'denial of service', information disclosure, and loss of information integrity.

There were 50 'Shellcode x86' alarms, and this alarm is described in the approach 1. Out from 125 alarms, 58 of them had the message 'ICMP Destination Unreachable Port Unreachable'. An ICMP port unreachable may indicate that someone or something tried to connect to a port on a system that was not available.

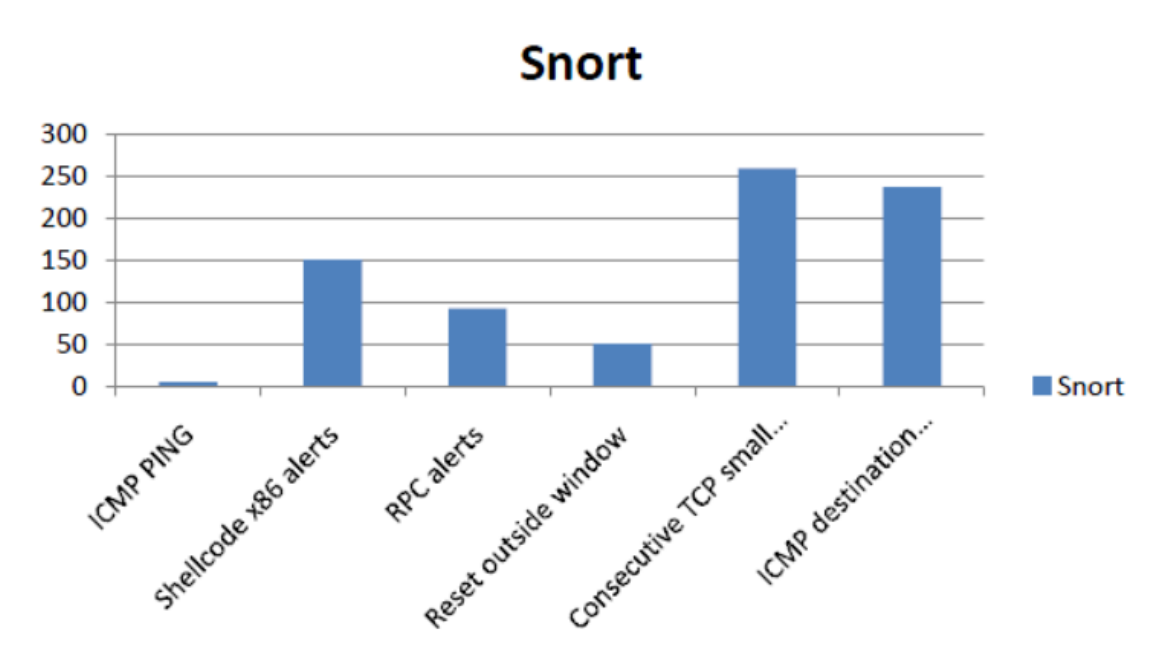
4.2. APPROACH 2

TEST 2



After running the Metasploit exploits against the target machine three times, capturing the traffic by tcpdump, and run Snort against this file, Snort created 752 alarms. 151 of them had priority 1, 358 of them with priority 2, and 243 with priority 3.

4.2. APPROACH 2



There were six 'ICMP PING' messages. An ICMP echo request is used by the the ping command to elicit an ICMP echo reply from a listening live host.

It was 151 different 'SHELLCODE x86' alarms, and 93 different RPC alarms. The 'Reset outside window' occur 51 times, and by looking up this alarms, it seems that it is a false positive, which mean that there is legitimate traffic that has been alarmed.

'Consecutive TCP small segments exceeding threshold' appeared 259 times, and there were 237 'ICMP detination unreachable' alarms.

4.2.2 Bro

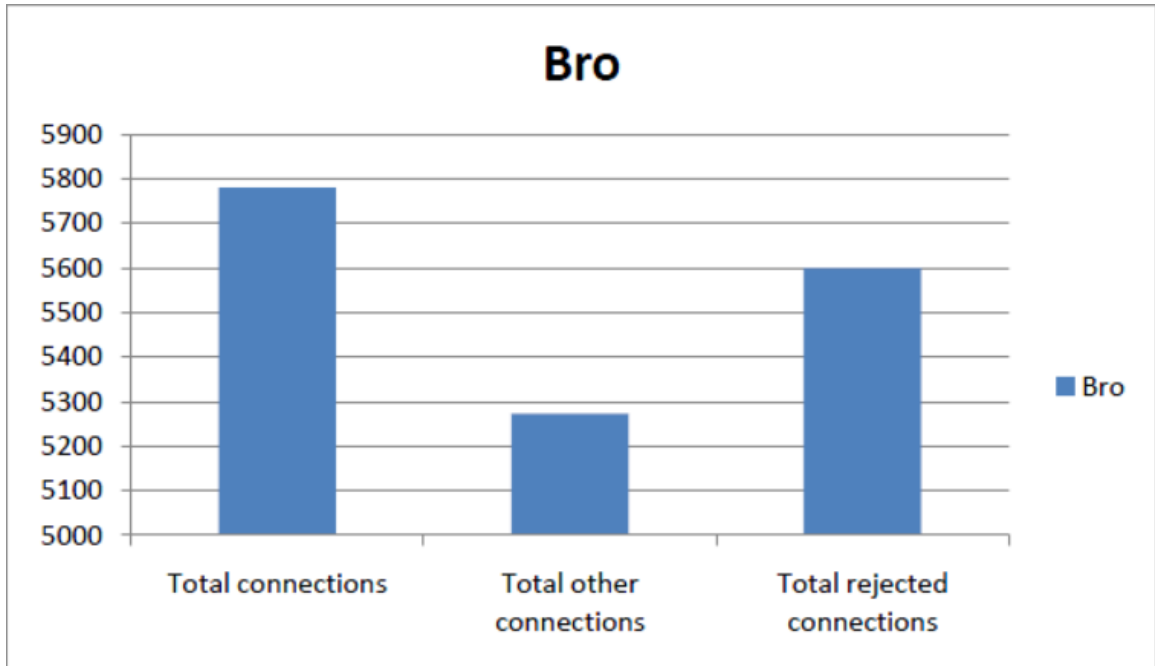
When connecting the laptop directly to the machine where Bro were installed, the Metasploit were run against the machine, and the traffic were captured by tcpdump into a file that ended with a size of 1,5 MB. After running Bro against this tcpdump file, it created 12 different logs as before, but some of them were empty. Those who were not empty, were 'alarm.log', 'conn.log', 'ftp.log', 'http.log', 'notice.log' and 'weird.log'.

In the **alarm log** one can information about how many hosts and ports that has been scanned, and other information. The alarms created in the alarm log are shown below:

- SensitivePortmapperAccess
pm-getport: sadmind
this alarm tells there were an attempt to gain administrator privilege
- PortScan
tells how many ports that has been scanned
a total of 300 ports were scanned
- LowPortTrolling
- PortScanSummary
tells how many ports that has been scanned
a total of 264 ports were scanned

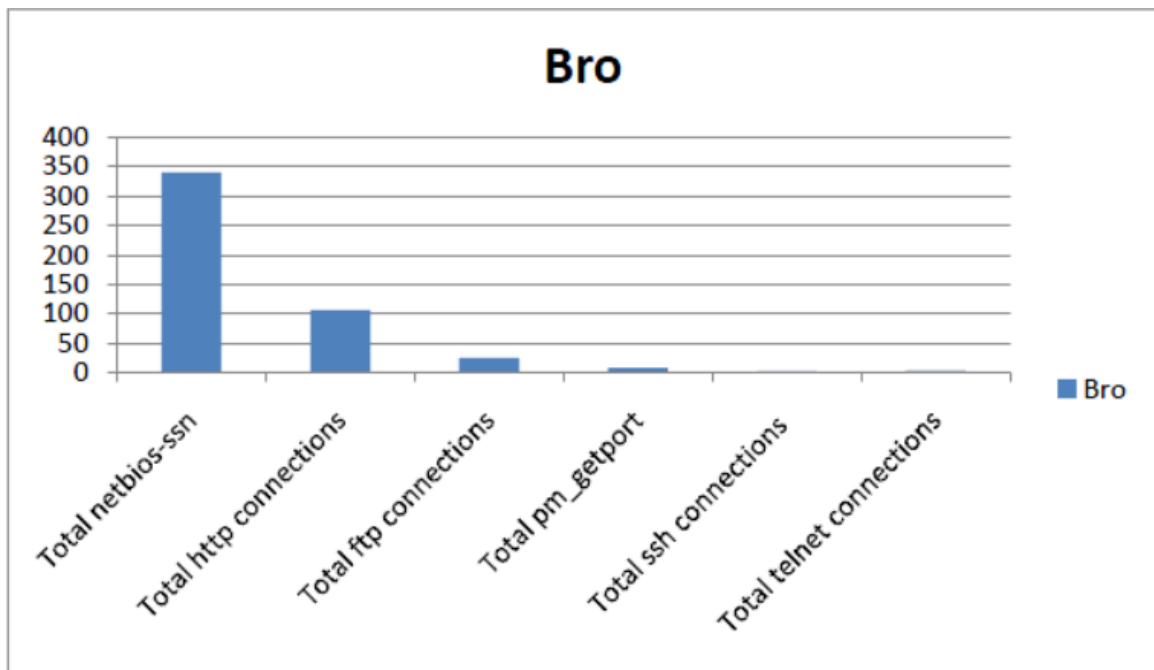
4.2. APPROACH 2

The connection log, log information about each connection. Timestamp, duration of the connection, source and destination IP, which ports that is involved and which protocol is logged.



Total connections in the connection log are 5781, and 5599 of the connections were rejected. Of the 5781 connections, 5273 of them did not have any specified name of the connection, as for example ftp or http or etc.

4.2. APPROACH 2



The rest of the connections in the connection log are listed here. There were a total of 340 netbios-ssn connections, 106 http connections, and 25 ftp connections. Total pm-getport connections were 8, it was 2 ssh connections, and 3 telnet connections.

There were only one http request in the http log:

```
1304592837.268426 %1 GET
/scripts/..5c..5cwinnt/system32/
cmd.exe?/x+/c+copy \winnt\system32\
cmd.exe x9WY.exe <no reply>
```

Some of the information that can be found in the notice log are similar to the information found in the alarm log. In the notice log, one can find timestamp, notice (alarm message) and notice action. Rest of the information differs because of the different alarms.

Alarms or notices listed in notice log:

- SensitivePortmapperAccess: 1 time
- OfflineResourceStats: 58 times
- PortScan: 2 times
- LowPortTrolling: 1 time
- ResourceSummary: 1 time

4.2. APPROACH 2

- PortScanSummary: 1 time

The last log containing information from the first tcpdump file, is the weird log. The weird log contains unusual events based on the weird policy. It represents packets that Bro consider suspicious.

```
1304592784.809614 10.0.0.2/47163 > 10.0.0.3/3000:  
above_hole_data_without_any_acks
```

```
1304592788.899341 10.0.0.2/52588 > 10.0.0.3/http:  
double_%_in_URI
```

```
1304592788.899341 10.0.0.2/52588 > 10.0.0.3/http:  
unescaped_special_URI_char
```

4.2. APPROACH 2

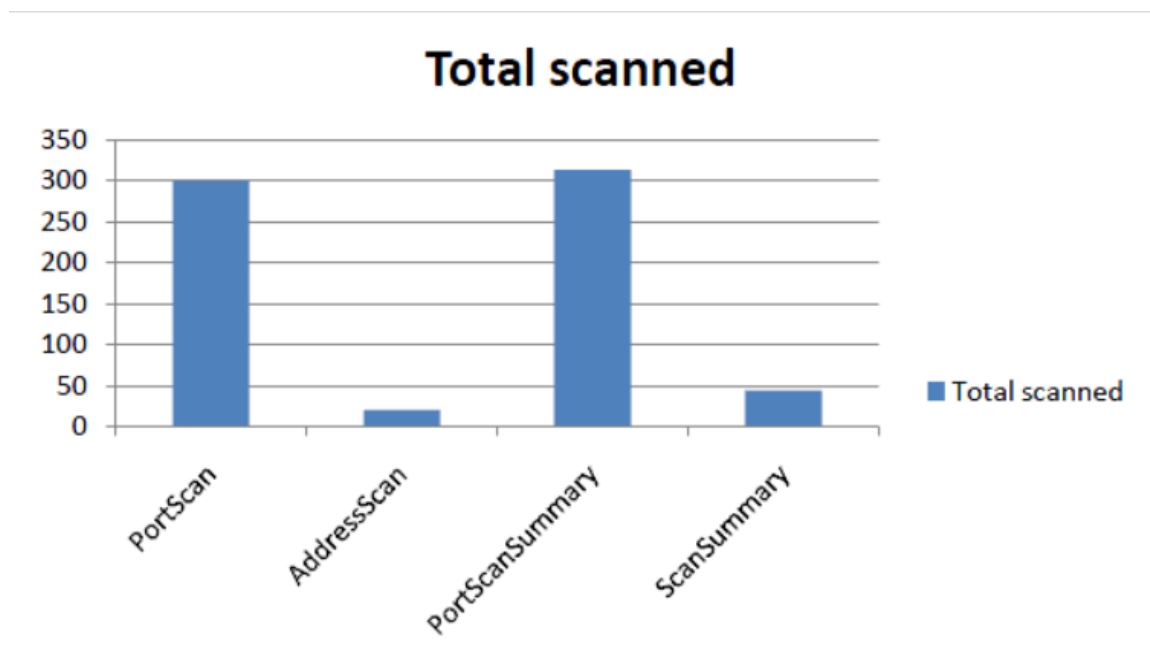
TEST 2

The second Metasploit approach, were to run it through the network, and capture the traffic with tcpdump and save it to a file. After running the same exploits three times, the tcpdump file ended with a size of 14 MB.

After running Bro against this tcpdump file, it was only four of the 12 logs that contained information. The logs with information was the 'alarm log', the 'conn log', and the 'weird log'.

The alarms or the scan summaries found in the alarm log after the second Metasploit test, are listed below:

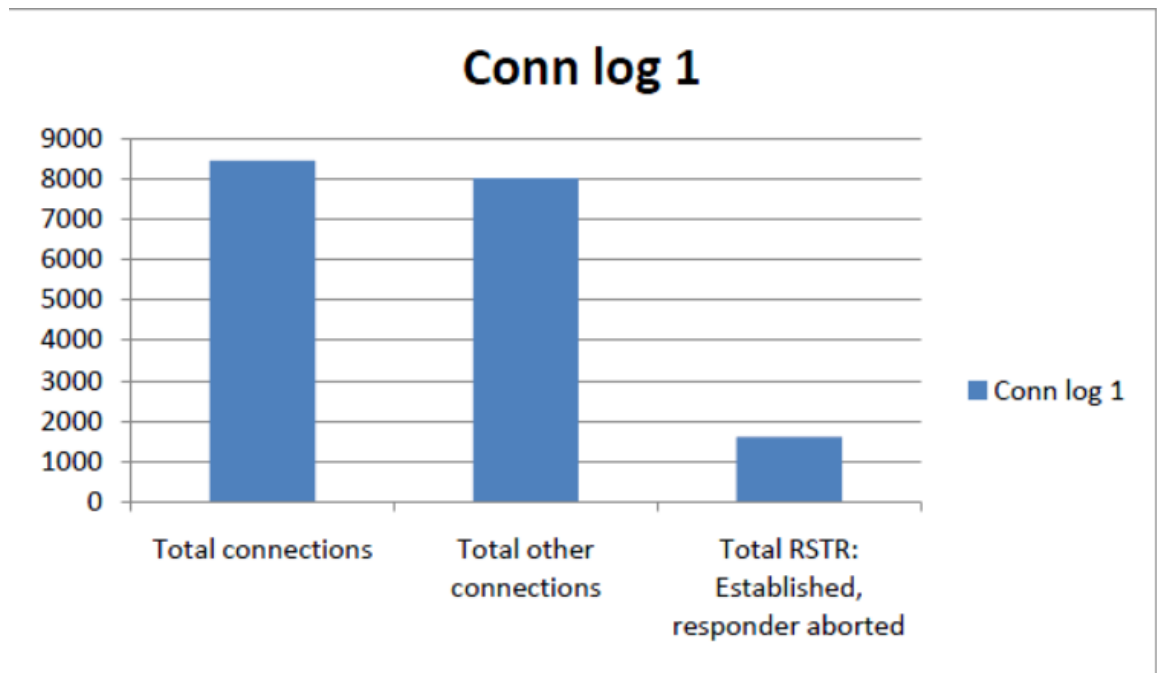
- SensitivePortmapperAccess
- PortScan
- AddressScan
- ScanSummary
- PortScanSummary



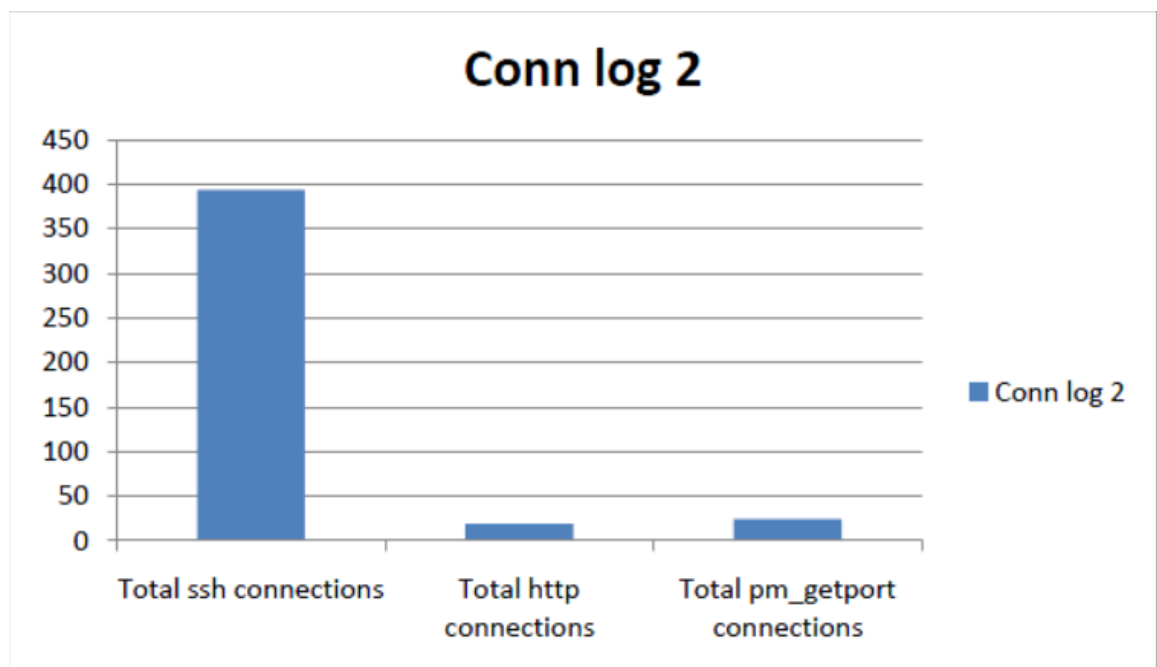
Portscan and Portscan summary tells how many port that have been scanned. Portscan shows that 300 ports have been scanned, while portscan summary shows that 313 ports have been scanned.

Addressscan and scan summary tells how many hosts that have been scanned, which are 20 hosts and 44 hosts.

4.2. APPROACH 2



Total of connections are 8451, and 8008 of the connections don't have any specified protocol used, as for example ssh, http and pm-getport that is shown in the next graph. 1611 of the connections were aborted by the responder.



394 of the connections were related to ssh, 19 to http, and 24 to pm-getport.

4.2. APPROACH 2

The same notices occurred in this notice log as the previous approach:

- OfflineResourceStats: 120 times
- SensitivePortmapperAccess: 1 time
- PortScan: 3 times
- TRWScanSummary: 1 time
- ScanSummary: 3 times
- PortScanSummary: 1 time

In the weird log there is four weird messages, where the IP of the laptop tried to connect on the port 3000 on the target machine from different ports. The message tells 'above hole data without any acks'.

```
1304697381.780930 128.39.75.86/51630 >
128.39.73.9/3000: above_hole_data_without_any_acks
1304697742.852886 128.39.75.86/42727 >
128.39.73.9/3000: above_hole_data_without_any_acks
1304697752.274955 128.39.75.86/41413 >
128.39.73.9/3000: above_hole_data_without_any_acks
1304698103.790528 128.39.75.86/39845 >
128.39.73.9/3000: above_hole_data_without_any_acks
```

4.2. APPROACH 2

4.2.3 Suricata

When running Suricata against the first tcpdump file, there were no alerts produced in the alert-debug.log or the fast.log. The http log and stats.log contained only a few lines about some http requests and some information about the packets in the tcpdump file.

Six http request were found in the http log, and in the stats log there were show how many packets that were processed

http request:

```
05/05/11-10:58:33.597680 10.0.0.3 [**]  
/_vti_bin/_vti_aut/fp30reg.dll [**]  
<useragent unknown> [**] 10.0.0.2:3000 -> 10.0.0.3:57781
```

Stats log:

13/5/2011 -- 12:23:42

Counter	TM Name	Value
decoder.pkts	Decode & Stream	13100
decoder.bytes	Decode & Stream	1267871
decoder.ipv4	Decode & Stream	12933
decoder.ipv6	Decode & Stream	0
decoder.ethernet	Decode & Stream	13100
decoder.raw	Decode & Stream	0
decoder.sll	Decode & Stream	0
decoder.tcp	Decode & Stream	12709
decoder.udp	Decode & Stream	163
decoder.icmpv4	Decode & Stream	58
decoder.icmpv6	Decode & Stream	0
decoder.ppp	Decode & Stream	0
decoder.pppoe	Decode & Stream	0
decoder.gre	Decode & Stream	0
decoder.vlan	Decode & Stream	0
decoder.avg_pkt_size	Decode & Stream	96.784046
decoder.max_pkt_size	Decode & Stream	1514
defrag.ipv4.fragments	Decode & Stream	75
defrag.ipv4.reassembled	Decode & Stream	3
defrag.ipv4.timeouts	Decode & Stream	0
defrag.ipv6.fragments	Decode & Stream	0
defrag.ipv6.reassembled	Decode & Stream	0
defrag.ipv6.timeouts	Decode & Stream	0
tcp.sessions	Decode & Stream	5767

4.2. APPROACH 2

<code>tcp.ssn_memcap_drop</code>	Decode & Stream	0
<code>detect.alert</code>	Detect	0

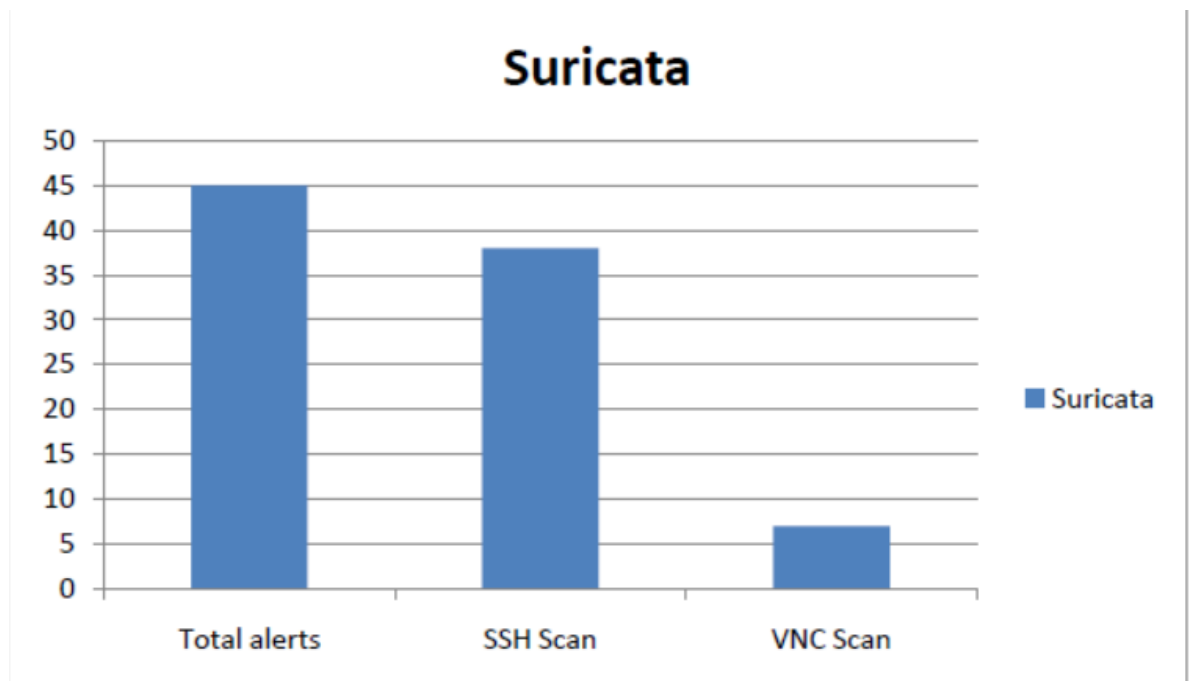
4.2. APPROACH 2

TEST 2

After running Suricata against the second tcpdump file, which contained traffic from three Metasploit exploits attack, it produced some alarms in the alert-debug log.

The same alarms can be found in the alert-debug log and the fast log. The results below are gathered from the fast log.

In the fast log one can find information, such as timestamp, such as timestamp, rule id, alarm message, alarm classification, which IPs and ports involved, some information about the alarm, and link to rule that triggered this specific alarm.



Suricata produced 45 alarms after reading the second tcpdump file. Of these 45 alarms were 38 of them potential SSH scan alarms, and 7 were potential VNC scans. These were the only two alarms Suricata created.

Some http requests from the second Metasploit test is listed below.

```
05/06/11-15:51:09.620756 128.39.73.9 [**]  
/pajax/pajax/pajax_call_dispatcher.php [**]  
<useragent unknown> [**] 128.39.75.86:38497  
-> 128.39.73.9:3000
```

```
05/06/11-16:02:32.273314 128.39.73.9 [**]  
/ [**] Mozilla/4.0 (compatible; MSIE 6.0;
```

4.2. APPROACH 2

```
Windows NT 5.1) [**] 128.39.75.86:41413  
-> 128.39.73.9:3000
```

```
05/06/11-16:02:35.945287 128.39.73.9 [**]  
/plugins/editors/tinymce/jscripts/tiny_mce/  
plugins/tinybrowser/edit.php?type=file&folder=  
[**] <useragent unknown> [**]  
128.39.75.86:48796 -> 128.39.73.9:3000
```

```
05/06/11-16:03:41.934284 128.39.73.9 [**]  
/ [**] <useragent unknown> [**] 128.39.75.86:55400  
-> 128.39.73.9:3000
```

The stats log holds information about how many packets, bytes, how many of them IPv4 or IPv6, and etc. This information is logged every eight second.

Chapter 5

Analysis and Discussion

5.1 Introduction

The installation and configuration turned out to be very time consuming. During the installation and configuration, different errors and problems appeared, and one of the intrusion detection systems (IDS) were more troublesome than the others.

Following an installation guide of how to install Snort, Bro and Suricata brought more problems than expected. During the installation of the different packages and libraries, and while including them for the different IDSs, different errors appeared mostly because some libraries or packages were missing and needed to be installed.

The results from the two approaches used, are going to be analyzed and discussed. As well, some different choices that have been made, and some improvements for the different IDSs will be explained.

After running Snort, Bro and Suricata for a period of time, they were stopped because it looked they were not able to run simultaneously. Therefore, tcpdump was used to capture traffic into a file, which Snort, Bro and Suricata could be run against. This applies to both approaches.

At a later point, when trying to run Snort, Bro and Suricata at the same time, it turned out to be that this approach works. But after using tcpdump, not any new tests were done by this approach. Tcpdump were stopped after four days because of the disk space were limited.

The first Metasploit test created a tcpdump file of 1.5 MB, which were a little bit low. At the second Metasploit test the Metasploit exploits were run three times in order to create a larger tcpdump file, and more traffic to analyze. This tcpdump ended at 14 MB.

5.2 Snort

5.2.1 Installation and configuration

Including the needed packages into Snort, gave some different types of problems. Dnet header not found and problem with PCRE were two problems that occurred. Finding the right solution for the dnet took some time, since libdnet and libnet were already installed and it should have been working. After spending quite some time on this problem, a solution were found:

```
LD_LIBRARY_PATH=/usr/local/lib
export LD_LIBRARY_PATH
```

These two commands had to be run at each login to the machine, and instead of doing this, the two commands were added to the `/root/.bashrc` file, which are run automatically at each login.

The issue with dnet, was that the path to where libdnet are installed (`/usr/local/lib/`) is not in the shared path used by Snort. After troubleshooting with this problem, a problem with PCRE appeared. It turned out to be some problems with the newest version, and an older version was installed. The rest of the process went without any further problems.

By following the installation guide, it was straight forward to install Snort, if looking away from the 'dnet header not found' problem. This were the only problem that took some time.

5.2.2 Approach 1

It took Snort 53 minutes and 19 seconds to read through the tcpdump file of 40 GB, and create alarms based on the traffic in the file. Snort were set only to log in the default logs, which were in ASCII format and full alerts. Snort produced as many as 408 390 alarms, which is a quite large amount of alarms after analyzing traffic from four days. By looking at the different alarms in the full alert file, there were many of the same alarms that were triggered many times.

By finding how many times each each alarm occur in the alert file, and finding out what each of the alarm means, one can see if there are some false positives and if the amount of them can be reduced by setting a threshold.

The total amount of ICMP PING were 90 029, and an ICMP PING are used to determine live hosts in the network. The information from the ping can be used by an attacker in prior to launching attacks. A simple ping are innocent in it self, since it does not do any harm. But the information from the ping, can be used further in an attack. If this message appear many times, it is likely that someone is trying to find live hosts in your network, and one should set

5.2. SNORT

a limit of how many times somebody can send an ICMP PING within a time period.

The alarm message 'Shellcode x86' occurred 137 156 times, and by looking up the sid (snort id) of 'SHELLCODE x86 inc ecx NOOP', one can see the amount of false positives are quite high. The corrective action to this alarm, is to apply a non-executable user stack patch to your kernel. Secure programming/execution of a program check the destination host and service to verify if any buffer overflow vulnerability exists. [22].

One could limit the amount of 'ICMP PING' and 'Shellcode x86' alarms by filtering in the threshold.conf file. This could be done in three different ways:

1. Limit: Alert each alarm only once
2. Threshold: Alert only once during a time period
3. Both above

An example could be to create one 'ICMP PING' alarm each minute. If there are more than one 'ICMP PING' alarm triggered within a minute, there will still be only logged one alarm.

'Bad segment' alarms are signs of anomalous traffic. This alarm is triggered by traffic that is outside the preset baseline that describes what kind of traffic that is considered normal. The message do not tell what kind of traffic that occurred, only that it was outside this preset baseline. This baseline are rules which describes what is considered anomalous traffic in the preprocessor files.

A port scan is run to find out what kind of ports is open on a system. An example of a portscan tool is nmap. When there are no services on the ports that have been scanned, Snort create an alarm message which says 'ICMP Destination Unreachable Port Unreachable'. A portscan are often used by an attacker to find which ports that are open on a host or server, and to use this information to attack the service on the open ports.

5.2.3 Approach 2

Metasploit was tested in two different ways; laptop directly connected to the target machine, and through the network. When using this two different methods, nmap had to be run two times since the IP address were different in the two tests.

- Test 1: IP address were 10.0.0.3
- Test 2: IP address were 128.39.73.9

5.3. BRO

After running nmap against each of them, different exploits were listed. In the first test there were listed as many as 347 exploits, while at the second test there were listed 127 exploits.

After running Snort against the first tcpdump file, it created 125 alarms. Knowing that there were sent 347 exploits towards the machine, it showed that Snort detected around 36 per cent of them, which are very low. By looking in the msfconsole log, it showed that not all exploits were run.

'ICMP Destination Unreachable Port Unreachable' tells that there have been a port scan. A port scan are used to find out which ports that is open. Normally there is run some services on the open ports, and knowing this information can be useful for an attacker. Since 58 of the alarms were alarmed with this message, it shows that 58 of the exploits failed.

A disadvantage with this message, is that it do not tell what kind of exploits that have been sent against the machine. The exploits in the Metasploit framework are made to exploit a vulnerability in a service run at the target machine. When there are no services on those ports these exploits are trying to attack, the exploits are aborted or stopped.

The RPC portmap alarms shows that there have been different attempts to exploit known vulnerabilities in Solaris, denial of service attempts, and attempt to discover which port that runs cmsd. This shows that some of the exploits sent from Metasploit triggered some alarms at Snort.

The different 'RPC portmap' alarms are portscans, attempts of to gain administrator privilege and denial of service attacks. Here have Snort discovered some of the exploits sent towards the machine. One could look at the exploits that have been run towards the machine, and see if one could find which of the exploits that have been triggered with 'RPC portmap' message. This is unfortunately not done.

After running Snort against the second tcpdump file containing traffic from the Metasploit exploits, it created 752 alarms. Still there were many alarms in the full alert file with the message 'ICMP destination unreachable port unreachable'.

5.3 Bro

5.3.1 Installation and configuration

The process of installing and finding out how to run Bro, were more time consuming than wanted. The first problem that appeared when installing Bro, were that automake and libtool were not installed. After installing these two

5.3. BRO

packages, the next problem appeared, which said the three packages libnucurses, libssl and libmagic had to be installed. Most of the packages are listed in the installation guide, but the problems appeared before using this guide.

When the problem with libpcap occurred, it was not because it had not been installed, but because that the version installed were not compatible with Bro. Therefore another version had to be installed. The last problem that appeared with installing Bro, was the PCRE package. It turned out to be some problems with the newest version, so an older version were installed.

After fixing all the problems during the installation, Bro finally got installed. By looking at the installation guide, it said that Bro was run like this: `bro.rc` start from where this file is located. When trying to run Bro in this way, the error message 'mkdir: cannot create directory' appeared. The solution to this was to install `bro-lite` and `broctl`, and run Bro with this command `broctl start`.

The installation of `bro-lite` lead to that three configuration files were installed in the `/usr/local/bro/etc` folder. This configuration files were the `networks.cfg` which told which network to run at, `node.cfg` which told which interface to run against, and `broctl.cfg` which told which rules that were included.

The last details that needed to be done to make Bro work, was that the `bro.cfg` had to be copied from `/usr/src/bro-1.5.3/scripts` to `usr/local/bro/etc`, and the `bro.rc-hoosk.sh` had to be copied from `/usr/local/src/bro-cvs/bro-1.5.3.cvs/scripts` to `/usr/local/bro/etc`. After this was done, Bro finally were able to run.

5.3.2 Approach 1

It took Bro 27 minutes and 51 seconds to read through the 40 GB `tcpdump` file. This is almost half the time of what Snort used. If only looking at this result, Bro has a detection engine that is twice as powerful as the Snort engine. When looking at the alarms created, there were a total of 95 584 alarms in the notice log, which are little bit under a quarter of the total alarms Snort created.

When looking in the different logs created by Bro, there are very little information about specific malicious activity compared to Snort. The closest are the address scan summaries and port scan summaries in the alarm log and notice log. Here one find summaries of how many ports and hosts that have been scanned. Information about different alarms such as 'ICMP PING', 'Bad Segment', 'Shellcode x86' that are found by Snort, are not found in the different Bro logs.

Instead of finding and logging information about potential threats in one file such as Snort, Bro separates the traffic into different files based on what

5.4. SURICATA

kind of traffic there is. All http requests get logged in the http log, all connections are logged in the conn log and all ftp traffic are logged in the ftp log. Bro creates a total of 12 different logs, and it can be time consuming to go through all this logs to look for malicious activity. But a positive thing with separating the different traffic into different logs, is that if one wants to look at ftp or smtp traffic, there is made own logs for this kind of traffic.

5.3.3 Approach 2

After running Bro against the first tcpdump file containing traffic from 347 exploits sent against the target machine, Bro created 12 logs as usual, but only 6 of them containing some information about the traffic. By looking in the alarm log, the summary of portscan showed that 300 ports had been scanned. Knowing that there were sent 347 exploits towards the machine, and that some of the exploits failed to run, it shows that Bro probably registrated all or most of the exploits that were sent.

By looking at the other logs created, such as the connection log one can see that there were 5781 connections made by the 347 exploits. This means that each exploit sent many connections requests to the different ports.

When running Bro against the second file containing traffic from the Metasploit exploits, Bro detected that 300 ports were scanned. And in the connection log there were registrated 8451 connections from the exploits.

5.4 Suricata

5.4.1 Installation and configuration

Installing Suricata with the needed packages, were straight forward. After learning from the problems and mistakes when installing Snort and Bro, it was just following the installation guide. The only process that took some time, were downloading the different rules from Emerging Threats [23].

5.4.2 Approach 1

Suricata were set to log into four different logs. After being run against the 40 GB tcpdump file, detailed information about each alarm where logged in the alert-debug log, information about each alarm in fast log, http request were logged in the http log, and statistics are logged in the stats log.

Suricata created a total of 28 243 alarms out from the tcpdump file, which are quite lower than both Snort and Bro. This are probably a more realistic number of alarms for four days, than the number of what Snort and Bro created.

5.4. SURICATA

The total of potential ssh scans were as high as 17 499. A SSH scan alarm get triggered when different usernames and password are tried quite frequently against port 22. Suricata can limit the amount of potential ssh scan by setting a limit of how many times a username and a password can be tried within a period. If nothing is set, as in this case, all ssh login attempts will be registrated as potentially ssh scan.

The TeamViewer alarm have been triggered 6986 times, and are created because of a typo. 'TeamVieweer' should be 'TeamViewer'. By looking up this alarm message, there are quite little information about it. So if the alarm is just a typo, then this traffic are looked as harmless and as a false positive. But in some cases, TeamViewer can be used in Social engineering, by tricking someone to download the program, and get remote access to their machines.

Rest of the alarms appeared not as many times as those two above. The 'Modified Sipvicious user-agent detected' alarm were listed 483 times, while total different 'Modified Sipvicious' alarms were 966. This message can mean different things, such as ssh scan, that a worm has been detected and port scan, which means that this alarm have to be taken serious.

At the end of each alarm in the fast log, one can find links to different web sites where information about the alarm can be found. This is similar to Snort's ability to look up different Snort IDs.

The 'Policy RPD' alarms shows different port scans, and there are not so many of them. Port scan are used to locate services on servers or host. If services are found at ports, the attacker will try attack and exploit a vulnerability on this port to gain access to the system, which is not wanted by those who run these services.

The message 'Likely BruteForce attack' are related to the potential ssh scan message. Brute force is when many different combinations of keys are tried to break in or gain access to a system. The 'Attempted administrator privilege gain' are related to the potential ssh scan as well.

The stats log are useful if one want to find out how many packets that have been processed, how many of them were IPv4 and IPv6, which protocols and further. Else, there are nothing specific information about malicious activity, as one can find in the fast log.

5.4.3 Approach 2

Suricata produced no alarms after running against the first tcpdump file containing traffic from the 347 exploits. None of the rules were triggered after reading the tcpdump file. This was very odd, since Snort produced 125 alarms from the same file, and Bro registrated that 300 ports were scanned. It could be that the ruleset that were used, were not good enough, but the solution to why Suricata did not produce any alarms remain unknown.

5.5. CONCLUSION

After running Suricata against the second tcpdump file containing traffic from Metasploit, 45 alarms were triggered. Compared to Snort and Bro this number of alarms were quite low. 37 of these alarms were SSH scans, and 7 of them were 'ET SCAN Potential VNC Scan 5900-5920 ' which told that some of the exploits tried to break in through vnc.

5.5 Conclusion

5.5.1 Installation and configuration

During the installation of Snort, Bro and Suricata, the one that turned out to be the easiest to install and configure, were Suricata. By following the installation guide, Suricata were installed with the needed packages. Snort and Bro encountered some problems during the installation process. Different packages that were either missing, packages that were not compatible, or it turned out to be wrong version installed. Bro were clearly the one that brought most problems during the installation. Just finding out how to run Bro, were a time consuming process, since the normal way to run Bro did not work.

5.5.2 Approach 1 and 2

Running Snort, Bro and Suricata against traffic captured by tcpdump, showed some huge differences in the amount of alarms created. In the first approach, Snort created four times as many alarms than Bro, and over 14 times as many alarms as Suricata.

Snort created over 400 000 alarms after reading the tcpdump file of 40 GB. This amount of alarms are pretty high when knowing that so many alarms are created based on traffic from just four days. The same applies Bro, which created almost 100 000 alarms. Snort have a capability to set threshold to reduce some of the most created alarms. The two alarms that appeared most in the full alert file, 'Shellcode x86' alarms and ICMP PING, could be reduced significantly by setting up thresholds.

Snort were quite sensitive, but the sensitivity can be configured into the threshold configuration file. The alarm messages in the alert file are shown in a way that are easy to read, and the possibility to look up the different snort id's is a very good attribute.

The Bro engine were not as sensitive as the Snort engine, but it still created quite many alarms. And instead of gathering information about each possible intrusion in one file as Snort, Bro saved various traffic into different logs. When creating different logs such as http log, ftp log, weird log and further, there are easier to discover possible intrusions of the different types of attack. This is the main advantage with the logs created by Bro. But something that

5.5. CONCLUSION

were disappointing, were the lack of information when looking up the different messages shown in the logs.

The Suricata engine turned out to be quite slower than Snort and Bro, when running against the tcpdump file. Suricata created not even close as many alarms as Snort and Bro, and one reason for this is that only limited set of rules were downloaded.

Useful information can be found in the fast log, such as what kind of alarms that have been triggered. After each alarm in the fast log, one can find several links to different web sites, where one can find information about the specific alarm. This is something similar to Snorts Snortid, and are very useful for finding if the different alarms are either real threats or false positives.

Bibliography

- [1] Symantec Threat Report. *www.symantec.com/business/theme.jsp?themeid=threatreport*, 2011.
- [2] Snort ids. URL: *www.snort.org*, 2011.
- [3] Money spent on computer security. URL: *http://blog.comptia.org/2010/08/16/making-sense-of-security-breach-cost-numbers*, 2010.
- [4] Security in computing 4th edition. Charles P. Pfleeger and Shari Lawrence Pfleeger, 2006.
- [5] Malware. URL: *www.wikipedia.org/wiki/malware*, 2011.
- [6] Computer security. URL: *wikipedia.org/wiki/Computer_security*, 2011.
- [7] Firewall. URL: *http://www.vicomsoft.com/knowledge/reference/firewalls1.html*, 2011.
- [8] Firewall. URL: *www.wikipedia.org/wiki/firewall_(computing)*, 2011.
- [9] Firewall limitations. URL: *http://www.sandiegopchelp.com/firewall-limitations.html*, 2011.
- [10] Antivirus. URL: *http://en.wikipedia.org/wiki/Antivirus_software*, 2011.
- [11] Intrusion detection system. URL: *www.wikipedia.org/wiki/intrusion_detection_system*, 2011.
- [12] Intrusion detection system. URL: *http://www.intrusiondetectionsystem.org/*, 2011.
- [13] Snort ids. URL: *www.wikipedia.org/wiki/snort_(software)*, 2011.
- [14] Snort manual. URL: *http://www.scribd.com/doc/6777057/Snort-Manual*, 2011.
- [15] Bro ids. URL: *www.bro-ids.org*, 2011.
- [16] Suricata ids. URL: *http://www.openinfosecfoundation.org*, 2011.
- [17] Suricata ids. URL: *www.wikipedia.org/wiki/suricata_(software)*, 2011.

BIBLIOGRAPHY

- [18] Sourcefire. URL: <http://en.wikipedia.org/wiki/Sourcefire>, 2011.
- [19] Palo alto networks. URL: www.paloaltonetworks.com, 2011.
- [20] Metasploit project. URL: www.wikipedia.org/wiki/Metasploit_Project, 2011.
- [21] Metasploit framework. URL: www.metasploit.org, 2011.
- [22] Snort id. URL: www.snortid.com, 2011.
- [23] Emerging threat rules. URL: <http://rules.emergingthreats.net>, 2011.
- [24] Hp procure 1800-24g switch. URL: <http://cdn.procurve.com/training/Manuals/1800-MgmtCfg-July2009-59916275.pdf>, 2011.
- [25] Installation guide of snort, bro and. URL: <http://blog.securitymonks.com/2010/08/26/three-little-idsips-engines-build-their-open-source-solutions>, 2011.
- [26] Installing metasploit. URL: http://dev.metasploit.com/redmine/projects/framework/wiki/Install_Ubuntu, 2011.

Chapter 6

Appendix

6.1 How to configure the HP Procurve 1800-24g switch

The steps of configuring the switch are: [24]

1. Connecting a laptop to the switch with a network cable
2. Turn of the wireless of the laptop
3. Set the IP on the laptop to be 192.168.2.10
4. Enter this IP in the web browser at the laptop
5. Entering the configuration page
6. Press continue without username and password
7. Choose which port to mirror traffic from: port 16
8. Choose which port to mirror traffic to: port 2
9. Save changes

6.2 How to install Snort, Bro and Suricata

6.2.1 Needed packages and libraries

There is some libraries that needs to be installed to make Snort, Bro and Suricata to work properly. The libraries or packages are listed below. [25]

Package	Description	Required by
Autotools	The “autotools” consist of autoconf, automake, and libtool. These will likely be installed on your system. You need the autotools if you will be using source from the Bro’s Subversion repository. You will need to run autogen.sh after you check out the code. We will go through the steps below.	Bro and Suricata
BIND8 headers and libraries	Most OSs will have BIND installed by default. BIND (Berkeley Internet Name Domain) is an implementation of the Domain Name System (DNS) protocols.	Bro
Bison or Byacc	Most OSs will have bison installed by default. Bison is a general-purpose parser generator that converts an annotated context-free grammar into an LALR(1) or GLR parser for that grammar.	Bro and Suricata
Flex	Most OSs will have flex installed by default. Flex is a tool for generating scanners. A scanner, sometimes called a tokenizer, is a program which recognizes lexical patterns in text.	Bro and Suricata
Libdnet	Libdnet provides a simplified, portable interface to several low-level networking routines.	Snort
Libpcap	Most OSs will have libpcap installed by default. It is the packet capture library. You may need to install it with support large files (files large than 2G). If you have a Linux kernel, you will want to configure libpcap for PF_RING support.	Snort, Bro and Suricata
LibYAML	LibYAML is a YAML parser and emitter written in C that is used to parse the configuration file.	Suricata
PCRE	The PCRE library is a set of functions that implement regular expression pattern matching using the same syntax and semantics as Perl 5.	Snort

6.2. HOW TO INSTALL SNORT, BRO AND SURICATA

These libraries above have to be installed to make Snort, Bro and Suricata work. Bu there exist other libraries that can be installed as well, to improve their function. These are not required packages or libraries, but will offer extra functionality for each of them.[25]

Package	Description	Required by
GnuPG	Free implementation of the OpenPGP standard.	Bro and Suricata
libcap-ng	The libcap-ng library is intended to make programming with posix capabilities much easier than the traditional libcap library.	Suricata
LibGeoIP	Ability to determine the location of IP addresses.	Bro and Suricata (future)
Libmagic	Add ability to determine file types, as with the ftp analyzer.	Bro
Libnet	Libnet is a generic networking API that provides access to several protocols	Suricata
OpenSSL	Tough to image a system not having OpenSSL installed. It is needed to analyze ssh certificates by the HTTP analyzer and for encrypted Bro to Bro communication.	Bro
PF RING	PF_RING is a new type of network socket that dramatically improves the packet capture speed.	Snort, Bro and Suricata
zlib	Libz is a compression library. It is used for decompressing HTTP bodies by the HTTP analyzer, and for compressed Bro-to-Bro communication.	Bro
XML analyzer	The XML analyzer is highly-experimental code written by Tobias Kiesling. Installation of Xerces-C++ and XQilla are required to use the XML analyzer. The code allows you to be able to easily adjust analysis capabilities to specific XML data formats. Xerces-C++ is a validating XML parser written in a portable subset of C++. XQilla is an XQuery and XPath 2 library and command line utility written in C++.	Bro and Suricata (future)
libnetfilter_queue	libnetfilter_queue is a userspace library providing an API to packets that have been queued by the kernel packet filter.	Suricata
libnfnetlink	libnfnetlink is the low-level library for netfilter related kernel/userspace communication. It provides a generic messaging infrastructure for in-kernel netfilter subsystems (such as nfnetlink_log, nfnetlink_queue, nfnetlink_conntrack) and their respective users and/or management tools in userspace.	Suricata

6.2.2 Installation

The first packages and libraries to install, is the autotool packages autoconf, automake and libtool. This was installed in this way:

- Apt-get install automake
- Apt-get install libtool
- Apt-get install autoconf

Other packages that were installed by apt-get install were:

- Apt-get install bison
- Apt-get install flex
- Apt-get install libmagic-dev
- Apt-get install libssl-dev
- Apt-get install g++
- Apt-get install libncurses5-dev
- Apt-get install svn-buildpackage
- Apt-get install libncurses-dev
- Apt-get install dnet-common

All packages or libraries that is not installed by 'apt-get install', is installed in the **/usr/src** directory.

```
Libdnet
Wget http://prdownloads.sourceforge.net/
libdnet/libdnet-1.11.tar.gz?download
Tar xf libdnet-1.11.tar.gz
Cd libdnet-1.11
./configure
Make & make install
```

```
Libpcap
Wget http://www.tcpdump.org/release/libpcap-1.1.1.tar.gz
Tar xvzf libpcap-1.1.1.tar.gz
Cd libpcap-1.11
./configure
Make & make install
```

6.2. HOW TO INSTALL SNORT, BRO AND SURICATA

LibYAML

```
wget http://pyyaml.org/download/libyaml/yaml-0.1.3.tar.gz
tar xzf yaml-0.1.3.tar.gz
cd yaml-0.1.3
./configure
Make & Make check & Make install
```

PCRE

```
Wget http://downloads.sourceforge.net/project/pcre/pcre/8.10/pcre-8.10.tar.gz
Wget http://sourceforge.net/projects/pcre/files/pcre/8.10/pcre-8.10.tar.gz
gpg --verify pcre-8.10.tar.gz.sig pcre-8.10.tar.gz
tar xzf pcre-8.10.tar.gz
cd pcre-8.10
./configure --prefix=/usr/local/pcre
Make & Make test & make install
```

Libcap-ng

```
wget http://people.redhat.com/sgrubb/libcap-ng/libcap-ng-0.6.4.tar.gz
tar xzf libcap-ng-0.6.4.tar.gz
cd libcap-ng-0.6.4
./configure
Make & make install
```

LibGeoIP

```
wget http://www.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz
gunzip GeoLiteCity.dat.gz
mkdir -p /usr/src/share/GeoIP
mv GeoLiteCity.dat /usr/src/share/GeoIP
wget http://www.maxmind.com/download/geoip/api/c/GeoIP.tar.gz
tar xzf GeoIP.tar.gz
cd GeoIP-1.4.6
./configure
Make & make check & make install
```

Libnet

```
Wget http://sourceforge.net/projects/libnet-dev/files/
libnet-1.1.5.tar.gz
Tar xf libnet-1.1.5.tar.gz
Cd libnet-1.1.5
./configure
Make & Make install
```

6.2. HOW TO INSTALL SNORT, BRO AND SURICATA

```
wget http://downloads.sourceforge.net/xqilla/XQilla-2.2.4.tar.gz
wget http://mirror.its.uidaho.edu/pub/apache/xerces/c/3/sources/
xerces-c-3.1.1.tar.gz
md5sum xerces-c-3.1.1.tar.gz
6a8ec45d83c8cfb1584c5a5345cb51ae  xerces-c-3.1.1.tar.gz
tar xzf xerces-c-3.1.1.tar.gz
tar xzf XQilla-2.2.4.tar.gz
ln -s XQilla-2.2.4 xqilla
cd xerces-c-3.1.1
./configure
Make & make check & make install
```

```
Google perftools checked..
Wget http://google-perftools.googlecode.com/files/
google-perftools-1.6.tar.gz
Tar xzf google-perftools-1.6.tar.gz
Cd google-perftools-1.6
./configure
Make & make check & make install
```

```
Libunwind: need to be installed to make perftools work.
Installed within the google perftool folder
Wget http://download.savannah.gnu.org/releases/
libunwind/libunwind-0.99-beta.tar.gz
Tar xzf libunwind-0.99.beta.tar.gz
Cd libunwind-0.99
./configure
Make & make install
```

```
OpenSSL
Were installed by apt-get install libssl-dev
```


6.2. HOW TO INSTALL SNORT, BRO AND SURICATA

6.2.3 Snort

Snort version 2.9.0.4 was downloaded from www.snort.org/snort-downloads and to the laptop.

Then it was copied to the machine "jonas.vlab.iu.hio.no (128.39.73.9)" with winscp, and put in the folder /usr/src.

Daq version 0.5 was downloaded from www.snort.org/snort-downloads and to the laptop. Then it was copied to the machine "jonas.vlab.iu.hio.no (128.39.73.9)" with winscp, and put in the folder /usr/src.

```
Tar xvzf daq-0.5.tar.gz
Cd daq-0.5
./configure --with-libpcap-includes=
/usr/local/include --with-libpcap-libraries=/usr/local/lib
```

In the /usr/src folder where Snort and daq were copied to:

```
Cat show_md5
"09b2a2d47d3de8106d0b625f7d8070c5"jonas:/usr/src#
```

```
Md5sum snort-2.9.0.4.tar.gz
09b2a2d47d3de8106d0b625f7d8070c5  snort-2.9.0.4.tar.gz
```

```
Tar xzf snort-2.9.0.4.tar.gz
Cd snort-2.9.0.4
```

```
CFLAGS="-D_LARGEFILE_SOURCE
-D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64"
LDFLAGS="-lpfring -lpcap"
LD_LIBRARY_PATH="/usr/lib:/usr/local/lib"
./configure --prefix=/usr/local/snort
--with-libpcap-includes=/usr/local/include
--with-libpcap-libraries=/usr/local/lib
--with-libpcre-includes=/usr/local/pcre/include
--with-libpcre-libraries=/usr/local/pcre/lib
--with-dnet-libraries=/usr/lib
--with-daq-libraries=/usr/src/lib
--enable-decoder-preprocessor-rules --enable-zlib
```

Because of some problems with dnet, these two commands were put in the /root/.bashrc file:

```
LD_LIBRARY_PATH=/usr/local/lib
```

6.2. HOW TO INSTALL SNORT, BRO AND SURICATA

```
Export LD_LIBRARY_PATH
```

```
In the /usr/src/snort-2.9.0.4 folder
Make & make check & make install
Mkdir -P /usr/local/snort/etc
Cp etc/* /usr/local/snort/etc
Mkdir -P /usr/local/snort/preproc_rules
Cp preproc_rules/* /usr/local/snort/preproc_rules
```

Rules

Snort can use their own set of rules from snort.org and rules from the Emerging Threats site. One needs to register at snort.org to get the rules from snort site.

The snort rules "snortrules-snapshot-2904.tar.gz" were downloaded from www.snort.org/snort-rules to the laptop and copied to the machine "jonas.vlab.iu.hio.no (128.39.73.9)" through winscp.

```
Md5sum snortrules-snapshot-2904.tar.gz
aa8e66072fcea05c25a78a834c220a  snortrules-snapshot-2904.tar.gz
```

```
mv snortrules-snapshot-2904.tar.gz /usr/local/snort/rules
tar xzf snortrules-snapshot-2904.tar.gz
```

Rules from emergingthreats.net were put into the /usr/local/snort/rules folder:
<http://www.emergingthreats.net/rules/emerging-all.rules>

6.2.4 Bro

```
wget ftp://bro-ids.org/bro-1.5.3-release.tar.gz
tar xzf bro-1.5.3-release.tar.gz
cd bro-1.5.3
```

Subversions trunk: in the /usr/local/src folder
Mkdir bro-cvs
Cd bro-cvs
svn checkout <http://svn.icir.org/bro/trunk/bro>
mv bro bro-1.5.3.cvs
cd bro-1.5.3.cvs

6.2. HOW TO INSTALL SNORT, BRO AND SURICATA

```
./autogen.sh
```

```
Make & make install
```

```
Configure and Install
```

```
From the "/usr/local/src/bro-cvs/bro-1.5.3.cvs" folder
```

```
CLAGS="-D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE  
-D_FILE_OFFSET_BITS=64" LDFLAGS="-lpfring -lpcap"  
LD_LIBRARY_PATH="/usr/lib:/usr/local/lib"  
./configure --prefix=/usr/local/bro --enable-debug  
--enable-perftools
```

```
Make
```

```
Make check
```

```
Make install
```

```
Brolite and Broctl:
```

```
Brolite and Broctl were installed in the  
/usr/local/src/bro-cvs/bro-1.5.3.cvs" folder by these commands:
```

```
Make install-brolite
```

```
Make install-broctl
```

```
/usr/local/bro/etc/node.cfg
```

```
Type: standalone
```

```
Host=localhost
```

```
Interface=eth2
```

```
/usr/local/bro/etc/networks.cfg
```

```
128.39.73.0/24
```

```
Two files needed to be copied to /usr/local/bro/etc:
```

```
- Bro.rc-hooks.sh needed to be copied from
```

```
/usr/local/src/bro-cvs/bro-1.5.3.cvs/scripts
```

```
- Bro.cfg needed to be copied from /usr/src/bro-1.5.3/scripts
```

6.2.5 Suricata

```
In the /usr/src folder:
```

```
wget http://www.openinfosecfoundation.org/download/  
suricata-1.0.2.tar.gz.sig
```

```
wget http://www.openinfosecfoundation.org/download/  
suricata-1.0.1.tar.gz
```

```
wget http://www.openinfosecfoundation.org/download/OISF.asc
```

6.2. HOW TO INSTALL SNORT, BRO AND SURICATA

```
gpg --import OISF.asc
gpg --verify Suricata-1.0.2.tar.gz.sig Suricata-1.0.2.tar.gz

tar xzf Suricata-1.0.2.tar.gz
cd Suricata-1.0.2

mkdir /usr/local/suricata

LD_RUN_PATH="/usr/lib:/usr/local/lib" ./configure
--with-libpcap-libraries=/usr/local/lib
--with-libpcap-includes=/usr/local/include/
--enable-nfqueue--enable-unittests
--enable-unified-native-timeval
--enable-profiling --prefix=/usr/local/suricata/

/usr/src/Suricata-1.0.2
Make & make check & make install

Mkdir /usr/local/Suricata/log
Mkdir /usr/local/Suricata/etc
Cp classification.config Suricata.yaml /usr/local/Suricata/etc
Mkdir /usr/local/Suricata/rules
Cd /usr/local/Suricata/rules
```

These rules were downloaded from the Emerging threats site:
<http://rules.emergingthreats.net>

```
emerging-activex.rules
emerging-policy.rules
emerging-attack_response.rules
emerging-pop3.rules
emerging-chat.rules
emerging-rpc.rules
emerging.conf
emerging-scada.rules
emerging-current_events.rules
emerging-scan.rules
emerging-deleted.rules
emerging-shellcode.rules
emerging-dns.rules
emerging-smtp.rules
emerging-dos.rules
emerging-snmp.rules
emerging-exploit.rules
emerging-sql.rules
```

6.3. HOW TO RUN SNORT, BRO, SURICATA AND TCPDUMP

```
emerging-ftp.rules
emerging-telnet.rules
emerging-games.rules
emerging-tftp.rules
emerging-icmp_info.rules
emerging-trojan.rules
emerging-icmp.rules
emerging-user_agents.rules
emerging-imap.rules
emerging-virus.rules
emerging-inappropriate.rules
emerging-voip.rules
emerging-malware.rules
emerging-web_client.rules
emerging-misc.rules
emerging-web_server.rules
emerging-mobile_malware.rules
emerging-web_specific_apps.rules
emerging-netbios.rules
emerging-worm.rules
emerging-p2p.rules
```

6.3 How to run Snort, Bro, Suricata and tcpdump

6.3.1 Snort

When running in the network, Snort is run like this:

```
/usr/local/snort/bin $:
snort -c /usr/local/snort/etc/snort.conf
-l /var/log/snort -i eth2
```

When running against a tcpdumfile in binary, Snort is run like this:

```
/usr/local/snort/bin $:
snort -c /usr/local/snort/etc/snort.conf
-l /var/log/snort
-r /root/jonas/testing/jonas_15aprilFull2.tcpdump
```

6.3.2 Bro

When running in the network, Bro is run like this:

```
/usr/local/bro/bin/ $: broctl start
```

When running against a tcpdump file, Bro is run like this:

6.4. HOW TO INSTALL METASPLOIT FRAMEWORK

```
/usr/local/bro/bin $:  
bro -r /root/jonas/testing/  
jonas_15aprilFull2.tcpdump brolite
```

6.3.3 Suricata

When running in the network, Suricata is like this:

```
/usr/local/suricata/bin $:  
suricata -c /usr/local/suricata/etc/suricata.yaml  
-s /usr/local/suricata/etc/classification.config  
-i eth2
```

```
/usr/local/suricata/bin $:  
suricata -c /usr/local/suricata/etc/suricata.yaml  
-s /usr/local/suricata/etc/classification.config  
-r /root/jonas/testing/jonas_15aprilFull2.tcpdump
```

6.3.4 Tcpdump

Tcpdump were run like this:

Approach 1:
tcpdump -i eth2 -s 0 -w jonas_aprilFull2.tcpdump

Approach 2:
tcpdump -i eth1 -s 0 -w test1

tcpdump -i eth2 -s 0 -w test2

6.4 How to install Metasploit framework

The Metasploit Framework were installed by following a guide at www.metasploit.org.
[26]

The Metasploit Framework were installed on a Ubuntu partition.

Downloaded the Metasploit framework
Wget [http://updates.metasploit.com/data/releases/
framework-3.6.0-linux-full.run](http://updates.metasploit.com/data/releases/framework-3.6.0-linux-full.run)

As root:
tar xf framework-3.6.0-linux-full.run
mkdir -p /opt/metasploit3

6.4. HOW TO INSTALL METASPLOIT FRAMEWORK

```
cp -a msf3 /opt/metasploit3
chown root:root -R /opt/metasploit3/msf3
ln -sf /opt/metasploit3/msf3/* /usr/local/bin
```

- apt-get install ruby
- apt-get install libopenssl-ruby
- apt-get install libyaml-ruby
- apt-get install libdl-ruby
- apt-get install libiconv-ruby
- apt-get install libreadline-ruby
- apt-get install irb
- apt-get install ri
- apt-get install rubygems
- apt-get install subversion
- apt-get install build-essential
- apt-get install ruby-dev
- apt-get install libpcap-dev
- apt-get install postgresql-8.4

Become the system postgres user

```
Sudo -s
```

```
Su postgres
```

```
postgres: createuser msf_user -P
```

```
enter password: 12345
```

```
enter it again: 12345
```

```
Shall the new role be a superuser? (y/n) n
```

```
Shall the new role be allowed to create databases? (y/n) n
```

```
Shall the new role be allowed to create more new roles? (y/n) n
```

```
Postgres: created -owner=msf_user msf_database
```

Configure Metasploit

```
Msf > db_driver postgresql
```

```
Db_connect msf_user:12345@127.0.0.1:5432/msf_database
```

```
Db_hosts ' list the new database and shows that it is empty,
```

6.4. HOW TO INSTALL METASPLOIT FRAMEWORK

since nothing is added yet

Enable the database on startup:

```
$ cat > ~/.msf3/msfconsole.rc
Db_driver postgresql
Db_connect msf_user:12345@127.0.0.1:5432/msf_database
Db_workspace -a MyProject
```

How to run Metasploit:

`nmap 128.39.73.9` : open ports are shown and saved to the created database

`db_autopwn -p -t` : lists exploits for these open ports that can be run. Exploits are listed below.

`db_autopwn -p -t -e` : the exploits listed get run.

Some of the exploits run against the target machine are listed below:

Matching Exploit Modules

128.39.73.9:3000 (port match)

- 1: exploit/unix/webapp/pajax_remote_exec
- 2: exploit/windows/misc/tiny_identd_overflow
- 3: exploit/windows/http/hp_nnm_nnmrptconfig_schdparams
- 4: exploit/windows/http/webster_http
- 5: exploit/windows/http/hp_nnm_ovwebsnmprsv_uro
- 6: exploit/multi/http/axis2_deployer
- 7: exploit/windows/http/shoutcast_format
- 8: exploit/unix/webapp/tikiwiki_jhot_exec
- 9: exploit/bsdi/softcart/mercantec_softcart
- 10: exploit/unix/webapp/oscommerce_filemanager

6.4. HOW TO INSTALL METASPLOIT FRAMEWORK

```
11: exploit/windows/http/maxdb_webdbm_get_overflow
12: exploit/windows/http/apache_mod_rewrite_ldap
13: exploit/unix/webapp/guestbook_ssi_exec
14: exploit/windows/http/hp_nnm_webappmon_execvp
15: exploit/windows/http/altn_webadmin
16: exploit/windows/http/hp_nnm_ovwebsnmprsv_ovutil
17: exploit/unix/webapp/mitel_awc_exec
18: exploit/unix/webapp/generic_exec
19: exploit/windows/http/hp_nnm_nnmrptconfig_nameparams
20: exploit/unix/webapp/awstats_migrate_exec
21: exploit/windows/isapi/ms03_051_fp30reg_chunked
22: exploit/windows/http/minishare_get_overflow
23: exploit/unix/webapp/base_qry_common
24: exploit/unix/webapp/mambo_cache_lite
25: exploit/unix/webapp/joomla_tinybrowser
26: exploit/windows/http/hp_nnm_ovalarm_lang
27: exploit/windows/http/newsms
28: exploit/windows/iis/ms03_007_ntdll_webdav
29: exploit/unix/webapp/dogfood_spell_exec
30: exploit/unix/webapp/phpmyadmin_config
```